Table of Contents

Reference Architecture (RA)

Return to Public Area **OR** View this document in the Wiki

DISTRIBUTED IMMUTABLE DATA OBJECT

(DIDO)

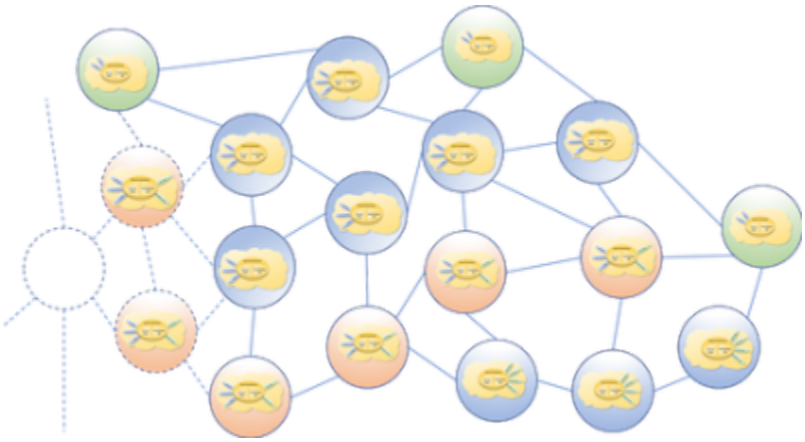
Reference Architecture (RA)

for

Cryptocurrencies, Blockchains, Distributed Ledgers and Tangles

Version 2.0

June 2020



R. W. “Nick” Stavros
Ian T. Stavros
Bryan Turek
Char Wales
Jackrabbit Consulting
Marc Abrams
John Black
Harmonia Holdings Group, LLC

Abstract

The excitement, expansion, and advancement in Distributed Computing resulting from Satoshi Nakamoto’s paper “Bitcoin: A Peer-to-Peer Electronic Cash System”[1] highlights the need for a Reference Architecture (RA) focused on Distributed Immutable Data Objects (DIDOs), where it is understood that DIDOs include, but are not limited to cryptocurrencies, the original blockchain, and distributed ledger technologies. This paper, an OMG Discussion paper, presents an RA for DIDOs.

- Front Matter
- Table of Contents

- DIDO-RA 2.0 PDF
- DIDO Data Moel (DIDO-DM) 1.0 PDF

1)

S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

Front Matter

return to Front of Book

- Cover Page
- OMG Disclaimer
- Summary of Changes
- Abstract
- Copyright
- Contents
- Preface

a. Cover Page

return to Reference Architecture (RA)

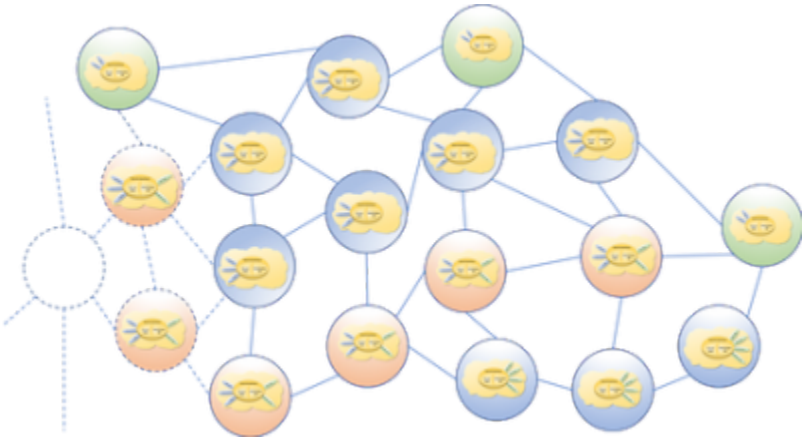
DISTRIBUTED IMMUTABLE DATA OBJECT
(DIDO)
Reference Architecture (RA)

for

Cryptocurrencies, Blockchains, Distributed Ledgers and Tangles

Version 2.0

June 2020



R. W. “Nick” Stavros
Ian T. Stavros
Bryan Turek
Char Wales
Jackrabbit Consulting
Marc Abrams
John Black
Harmonia Holdings Group, LLC

OMG Discussion Paper Disclaimer

return to Front Matter

In accordance with the OMG's Policies & Procedures (P&P), to be issued as an OMG Discussion Paper, the DIDO RA 2.0 must include the following Disclaimer:

> *This paper presents a discussion of technology issues considered in a Subgroup of the Object Management Group. The contents of this paper are presented to foster wider discussion on this topic; the content of this paper is not an adopted standard of any kind. This paper does not represent the official position of the Object Management Group.*

b. Summary of Changes from 1.0

return to Front Matter

The major change is moving the content from a Microsoft Word document to a Wiki. This transition allows for content to be accessed on a section by section basis, easy cross-referencing of each section, and the ability to easily determine not only how many times a section is referenced, but also where. Specific changes done or in process include:

- Technical section
  o Reorganized into a series of technical views, citing relevant technical & de facto standards for each
  o As appropriate, tools to support technical views: added
- Taxonomy of data and nodes: added to Technical Views
- Governance for managing the DIDO Architecture: added
- Glossary of Terms (Appendix A): added
- A new section on technical and de facto standards (Appendix B): added
  o Data sheets for technical standards: added
  o Data sheets for de facto standards: added
- A new section for tools (Appendix C): added

c. Abstract

return to Front Matter

The excitement, expansion, and advancement in Distributed Computing resulting from Satoshi Nakamoto's paper "Bitcoin: A Peer-to-Peer Electronic Cash System"[2] highlights the need for a Reference Architecture (RA) focused on Distributed Immutable Data Objects (DIDOs), where it is understood that DIDOs include, but are not limited to cryptocurrencies, the original blockchain, and distributed ledger technologies. This paper, an OMG Discussion paper, presents an RA for DIDOs.

2)

S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

d. Copyright Notice

return to Front Matter

f. Preface

return to Front Matter

The Distributed Immutable Data Objects (DIDO) Reference Architecture (RA) is meant to be used as a resource to guide the design, use, or selection of Blockchain, Distributed Ledger Technology (DLT), or other Distributed Computing solutions such as InterPlanetary File System (IPFS) and Data Distribution Service (DDS).

The purpose of DIDO RA 1.0 was to create a better understanding of the Blockchain and DLT, which were exploding in the Information Technology (IT) world after the publication of Satoshi Nakamoto's paper "Bitcoin: A Peer-to-Peer Electronic Cash System"[3] and the subsequent success of the Bitcoin. Since the publication of Nakamoto's paper, this excitement has grown way beyond the original Bitcoin. It has led to the promise/emergence of many other new cryptocurrencies, as well as the application of the well known and established concepts of distributed, peer-to-peer applications to supply chains, the Industrial Internet of Things (IIoT), natural resources, environmental sciences, and even the monetization of data.

In DIDO RA 2.0, the goal is to focus less on cryptocurrencies and more on generalizing peer-to-peer, distributed computing. As a parallel effort to the publication of DIDO RA 2.0, several products have been developed to work in parallel with and complement this paper:

- **DIDO Data Model (DIDO-DM)**: captures the conceptual data constructs described in the DIDO-RA including the Community of Interest (CoI) and testing.
- **DIDO Testing Environment (DIDO-TE)**: creates an environment that allows for virtualized testing of a Distributed Application (DApp) before it can be released into the "wild" using real hardware and networks.
- **DIDO Command Line Interface (CLI)**: defines a high level command language with which to send commands to each node covering the configuration, definition, and manipulation of data on distributed nodes.
- **DIDO Reference Implementation (DIDO-RI)**: provides a working interface to the DIDO-DM, DIDO-TE and DIDO-CLI.

A Special Thanks. This work would not have been possible without the efforts of:
• Peter Denno, Computer Scientist, National Institute of Standards and Technology (NIST), Engineering Laboratory, 100 Bureau Drive, Gaithersburg, MD 20899

3)
S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

1 Introduction

Return to Reference Architecture (RA)

The term Distributed Immutable Data Objects (DIDO) refers to the underlying technologies supporting distributed data and computation across a distributed network of peers using consensus algorithms to maintain integrity and consistency across the network. After the publication of Satoshi Nakamoto's paper *Bitcoin: A Peer-to-Peer Electronic Cash System*[4)] and the exponential growth of other cryptocurrencies, there is a need to understand in general terms the underlying DIDO architectures and provide a framework to enable engineering due diligence of DIDOs. DIDOs are not limited to cryptocurrencies, the original blockchain, or distributed ledger technologies. DIDOs are also applicable to other non-cryptocurrency domains such as supply chain, registries for births, deaths etc., and lists of acceptable Identification and Authentication (IA) acceptable certificates, including those that have been revoked. The DIDO concepts captured within a Reference Architecture (RA) are intended to represent any architecture relying on distributed networks of peers that store data and allow parallel computation. The DIDO RA is not intended as a physical "must-have" requirements list, but more as a "what-can-be" conceptual catalog.

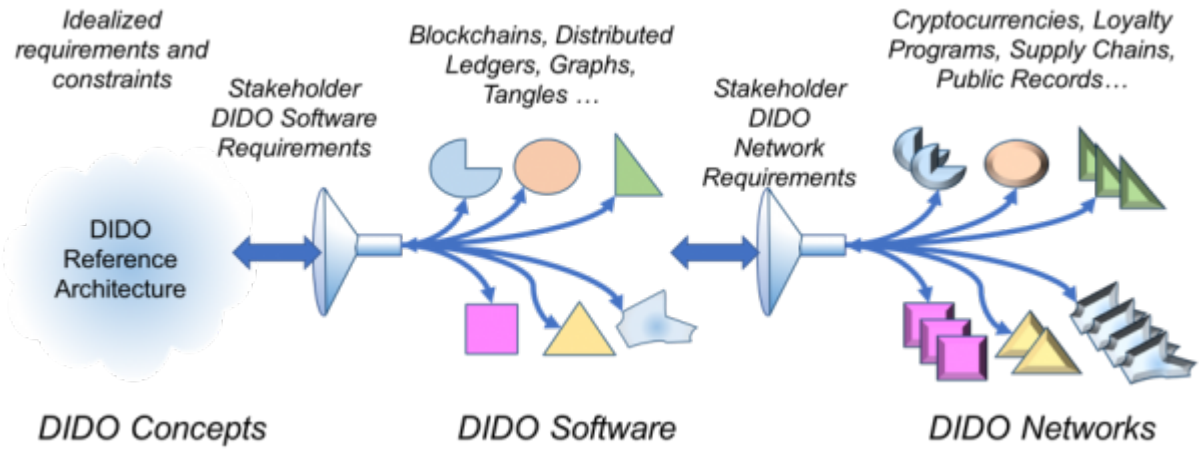


Figure 1: The relationship of DIDO Reference Architecture to incarnations such as Bitcoins, distributed ledgers, and tangles.

As illustrated in the figure, the DIDO RA is an idealized general set of requirements and constraints. Each incarnation of the DIDO software (e.g., cryptocurrency or distributed ledger) uses a set of stakeholder requirements to filter the DIDO RA and tailor it to suit the unique needs and desires of its community of stakeholders. For example, the DIDO RA provides standards for logging, even though the logging requirements driven by DIDO software stakeholders may lead to a decision to opt not to include them. There are a plethora of DIDO software incarnations available, starting with the original Blockchain software that drives Bitcoin, and moving onto IBM's Distributed Ledger, Ethereum, and Iota.

The DIDO software incarnations are adopted or used by another community of stakeholders that wish to leverage the DIDO software into a DIDO Network of Nodes to address requirements and needs of a specific domain. For example, one DIDO network community wants to provide a cryptocurrency and another public records. The DIDO software selected by the DIDO network stakeholders might be different depending on its intended purpose. In Figure 1, the bi-directional arrows communicate the notion that

the DIDO RA influences not only the DIDO software and networks, but also that evolution of DIDO software and networks feeds back into subsequent versions of the DIDO RA.

---

4)

S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

1.1 Problem

return to Introduction

As of June 12, 2018, there were 1,628 Cryptocurrencies available, with a total market cap around $300 Billion[5),6)]. DIDO implementations and interest in them are currently in the Positive Hype part of the technology Hype Cycle as defined by Gartner.[7)]

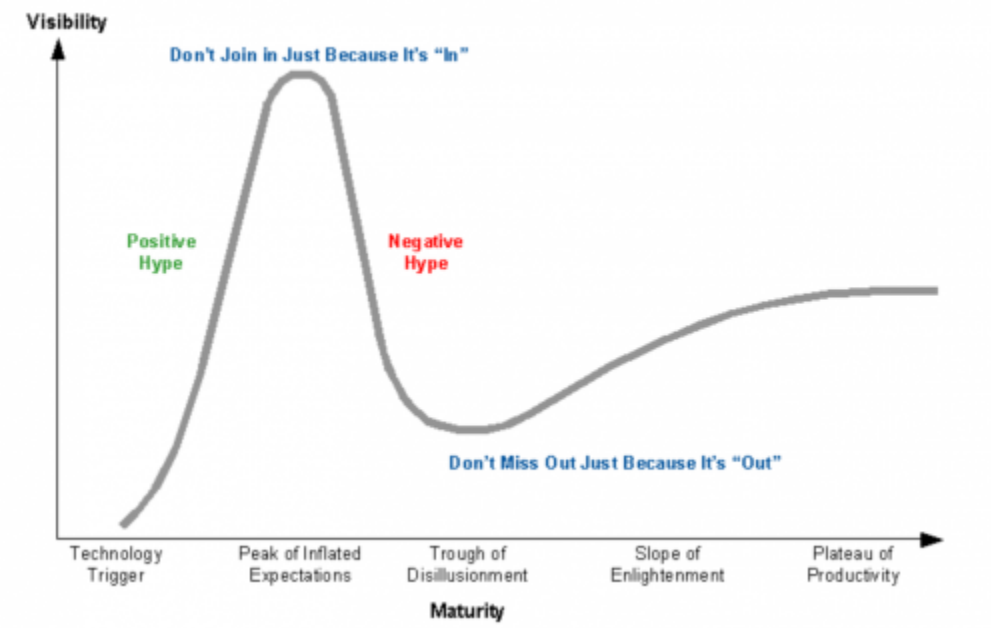


Figure 2: Gartner Group Hype Cycle

A major concern and part of the motivation behind the creation of a Reference Architecture (RA) is the lack of a comprehensive mechanism to evaluate all the risks associated with DIDO implementations (especially cryptocurrencies). Many of the risks to DIDOs are common to distributed computing and have already been identified and addressed using risk mitigators in existing processes, procedures, and standards. An example is vulnerabilities in source code. unfortunately, no statistics are available on how rigorously these risk mitigators are applied to DIDOs, if at all. Specifying a Reference Architecture and cross-referencing its components to a set of existing standards establishes the following:

- An actuarial framework for assessing risks associated with existing products
- Metrics for evaluating, comparing, and selecting existing products
- A roadmap for future standard implementation, enhancement, and development

5)

Coin Market Cap, "Cryptocurrency Market Capitalizations," 3 December 2017. [Online]. Available: https://coinmarketcap.com/all/views/all/. [Accessed 3 December 2017].

6)

D. Palmer, "Market Cap Hits All-Time High," 23 August 2017. [Online]. Available: https://www.coindesk.com/150-billion-total-cryptocurrency-market-cap-hits-new-time-high/. [Accessed 20 November 2017].

7)

Understanding Gartner's Hype Cycles, 30 May 2003, Alexander Linden, Jackie Fenn,

https://www.bus.umich.edu/KresgePublic/Journals/Gartner/research/115200/115274/115274.html

1.2 Purpose

return to Introduction

The goals of a Reference Architecture (RA) as expressed in this paper are derived from the Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII)[8)] , which are:

- Provide a common language for the various stakeholders
- Provide consistent implementation of technology to solve problems
- Support validation and comparison of implementations
- Encourage adherence to common standards, specifications, and patterns

Achieving these goals will enhance the likelihood that DIDOs and networks will be engineered correctly. DIDOs are intended to cover the entirety of distributed computing, including improvements in data storage and computing that are now used to support DIDO processes, but were not considered in Nakamoto's paper[9)] and the consequent highly successful launch of Bitcoin cryptocurrency. Nakamoto's paper proposed a "ledger" for storing data and a collection of transactions, captured in a block. Blocks of transactions are verified and validated using a consensus algorithm. Bitcoin relies on Proof-of-Work (PoW) consensus. This solution, although usable, has proven to be expensive, making widespread ubiquitous adoption difficult[10)]. Other implementations such as Ethereum were built upon the Bitcoin blockchain concept, which replaced the costly PoW with Proof of Stake (PoS). The DIDO RA is extensible to evolving and emerging blockchain technologies. For example, Boyen[11)] notes that:

> *Our blockchain-free proposal shifts onto the transactions themselves the task of affirming prior transactions. Verification no longer results in a chain of transactions blocks, but in a lean graph comprised only of transactions...* [12)]

The DIDO RA concerns distributed, Peer to Peer (P2P) computing including blockchains and cryptocurrencies, but also covers domains that have little or nothing to do with currencies. For example, DIDOs can be blockchains, distributed ledgers, or graphs. Blockchains may use any consensus algorithm such as PoW or PoS. Cryptocurrencies are used as a placeholder for any of the other domains that can be supported using DIDO: supply chains, government records, scientific data, medical records, escrows, swaps, etc.

The explosion in cryptocurrencies is well known and documented. An example is presented extremely well in the animation provided by Jeff Desjardins[13)].

> **Note** "ICO funds raised" went from zero in 2014 to about $6.5 billion in November 2017.

Figure 3: The explosive growth of Initial Coin Offerings (ICOs) over four years

To provide a common language for stakeholders, the DIDO RA describes the components of a distributed network of peers supporting distributed data and computation. It is comprised of a collection of peer nodes that operate within a virtual distributed network. Each node in the architecture selects the set of architectural components, and the relationships between the components, required by their stakeholders to solve the specific requirements (e.g., blockchains, cryptocurrencies, distributed ledgers, graph databases). The individual nodes synchronize via distributed software communicating over secure messaging infrastructure. All computations and operations could be executed redundantly on all the nodes within the DIDO network of peers. DIDO components are virtual representations of functionality found in DIDO products. In addition to identifying and defining the components and their inter-relationships, the RA associates each component with existing standards (refer to Section 2.1.7).

8)
G. Doyle and B. Wilezynski, “Reference Architecture Description,” U. S. DoD, Washington, DC, 2010.
9)
S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.
10)
S. Lee, “Bitcoin's Energy Consumption Can Power An Entire Country – But EOS Is Trying To Fix That,” 2018 Aoril 2018. [Online]. Available: https://www.forbes.com/sites/shermanlee/2018/04/19/bitcoins-energy-consumption-can-power-an-entire-country-but-eos-is-trying-to-fix-that/#7a0603931bc8. [Accessed 11 June 2018].
11)
Gartner, “Gartner Hype Cycle,” [Online]. Available: https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp#. [Accessed April 2018].
12)
C. C. T. H. Xavier Boyen, “Blockchain-Free Cryptocurrencies, A Framework for Truly Decentralized Fast Transactions,” December 2017. [Online]. Available: https://eprint.iacr.org/2016/871.pdf. [Accessed 17 December 2017].
13)
J. Desjardins, “Business Insider - Visual Capitalist,” 13 December 2017. [Online]. Available: http://www.businessinsider.com/animation-shows-the-explosion-in-ico-funding-over-the-last-four-years-2017-12. [Accessed 17 December 2017].

1.3 Content Organization

return to Introduction

Following this Introduction, this discussion paper is organized as follows:

Section 2, Architectural Views: Presents and describes in detail the DIDO Architecture as a set of components collected into complementary Architectural Views: Stakeholder, Technical, and Taxonomic.

- **The Stakeholder View** is organized in accordance with the communities considered to be the "customers" of the DIDO RA, e.g., Platform, Domain, and Ecosystem.
- **The Technical View** is organized into two basic views: Fundamental (e.g., System of Systems, Assurance, Tools) and Node Network.
- **The Taxonomic View** is organized into four taxonomies: network topology, network access control, nodes, and data.

Standards (technical and *de facto*) relevant to or associated with the components supporting each view are listed.

Section 3, Governance: Proposes and describes the elements of the governance structure to manage the DIDO Architecture as an Open Source program, i.e., Charter (for the effort as a whole plus each of the DIDO COIs), Policies & Procedures (P&P), and Guide (plain English version of the P&P).

Appendices These are links to the first page of each appendix. Links within these appendices go to the public DIDO Wiki pages.

- Appendix A: Glossary of Terms
- Appendix B: Standards Organizations: Complete listing and description of the Standards Organizations associated with the technical and *de facto* standards cited in the DIDO RA Architectural Views.
- Tools
- Acronyms

2 Architectural Views

return to Reference Architecture (RA)

The DIDO Reference Architecture (RA) is presented as a series of components collected into different views. The components are then associated with a set of standards relevant to each component. There are two kinds of standards provided, each produced by different kinds of standards bodies:

- Technical Standards Bodies
- de facto Standards Bodies

There are three main categories of Architectural Views:

- 2.1 Stakeholder Views
- 2.2 Technical Views
- 2.3 Taxonomic Views

2.1 Stakeholder Views

return to Architectural Views

Within this context, the following definition of Stakeholder is used:

> *A **stakeholder** is a person, group or organization that has interest or concern in a* [target] *organization. Stakeholders can affect or be affected by the* [target] *organization's actions, objectives and policies. http://www.businessdictionary.com/definition/stakeholder.html*
> **Note**: [target] added for clarification purposes

In a centralized or decentralized topography, this definition is adequate; however, in a distributed topography the understanding of what the *[target]* organization is becomes important. In a DIDO, this is by design and intent. There is no centralized authority or centralized cluster for the data, the processing of which is considered a major feature of the distributed architecture. It is a network of peers working together in parallel and simultaneously to solve problems. In other words, no single organization owns:

- all the computer resources or control them
- the definition of, or has control over, the processes
- the definition of data structures or data being distributed

In contrast to distributed systems, centralized systems (i.e., mainframes) are the authority for computation and data. In essence, the only reality is the processes and data that reside in the centralized system.

This is also in contrast to decentralized systems (i.e., traditional cloud servers), which rely on well-orchestrated and coordinated efforts of a few well connected and synchronized systems. Collective servers are the authority for the computation and data. In essence, the only reality is that which can be found on the decentralized servers; the infrastructure is expected to keep the software and data consistent and synchronized. However, this only needs to be concerned with servers that, although they are decentralized, still fall under a single authority thereby making the requirements, architecture, design, implementation, and maintenance relatively easy.

Both centralized and decentralized systems often have extensive data models and functionality, which adds to the complexity of managing them. This generally requires a single governing body (i.e., enterprise) to be ultimately responsible for the entire ecosystem and the lifecycle of the systems and the integration of components including hardware, operating systems, database management systems, web servers, application servers, software languages, networking, and other protocols. These “stacks” often result in stovepipe solutions.

In distributed systems, much of the ecosystem and governance of the components is handled by various Communities of Interest (CoIs): each has a responsibility for different aspects of the distributed system. The traditional role of a corporation or enterprise is to participate in these communities. The following graphic illustrates the various communities considered to be “customers” of the DIDO RA.

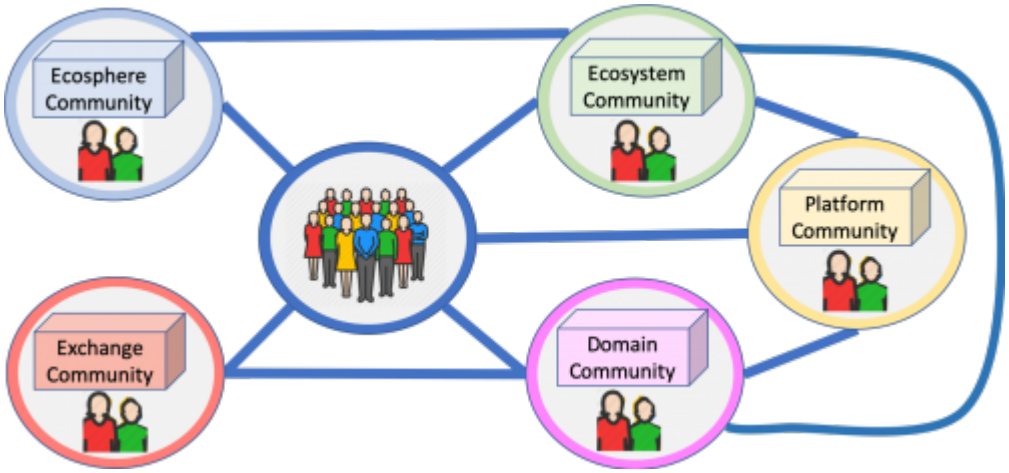


Figure 4: DIDO RA Stakeholder Communities

Platform

is responsible for the software used to distribute and control data within the Node Network, for example Bitcoin, Ethereum, Iota, DDS, and IPFS.

Domain

is responsible for the use of the data distributed on the Node Network, for example currency, rewards programs, or certificates.

Ecosystem

is responsible for a collection of domains associated with a particular ares of interest, such as green groceries, interest rate swaps, a particular tank, or class of automobiles.

Ecosphere

is responsible for a collection of domains and ecosystems associated with a common governance, which crosses over multiple areas of interest, such as military, government, automotive, or finance.

Exchange

is responsible for the exchange of data (tokens) from one domain or ecosystem with data in another domain or ecosystem, for example exchanging Bitcoins for U.S. dollars, or strawberries for jars of jam.

Enterprise

is responsible for being the systems integrator of all the domains, ecosystems, and ecospheres needed to fulfill the mission and goals of a corporation or organization, such as

an auto company, a chain of retail stores, or a bank.

Each of these areas is explained in more detail in the following views, concluding with a list of standards applicable to these Stakeholder Views:

2.1.1 Platform View

return to Stakeholder Views

A platform community of interest (CoI) is responsible for the DIDO infrastructure that resides on each node in the node network. Direct involvement within a platform covers the gamut from observers, to passive users of the software, and significant contributors to the community or de facto standards. As a general rule, domains, ecosystems, and ecospheres participate in platform communities rather than sponsor them. For the most part, platforms are already existing, self-contained communities with well established governance. Common CoI activities are bug tracking, collaboration and voting on resolutions of problems and future directions.

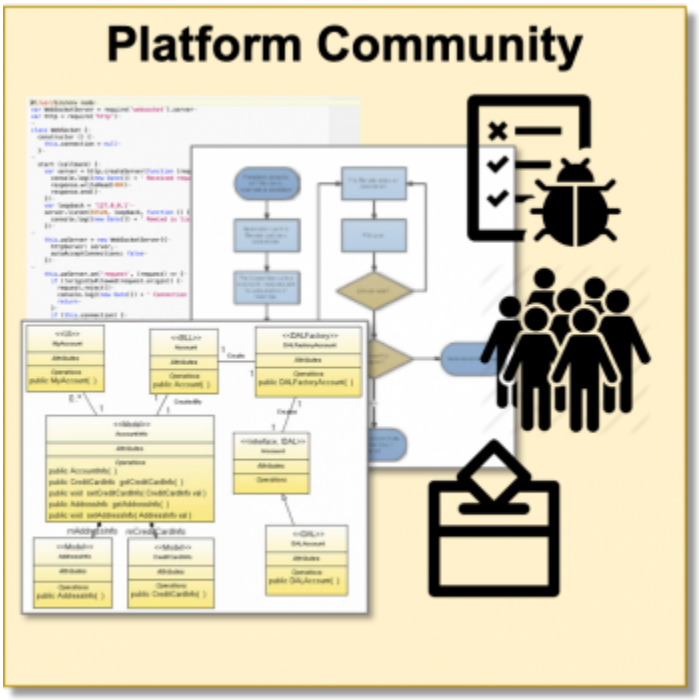


Figure 5: Platform Community

Some examples of platforms are:

- Bitcoin
- DDS Foundation
- Ethereum
- Hyperledger
- Iota
- IPFS
- Multichain

Standards

Technical Standards

- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)
- OMG: Interface Definition Language (IDL)

de facto Standards

- InterPlanetary File System (IPFS)
- Bitcoin
- Ethereum
- Linux Foundation: Hyperledger
- IOTA

2.1.2 Domain View

return to Stakeholder Views

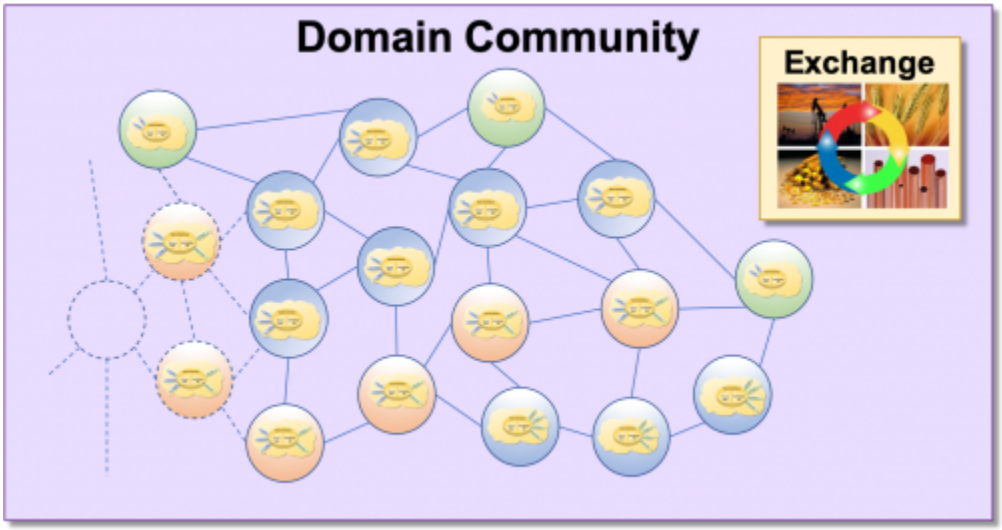


Figure 6: A Domain Community

A DIDO network consists of a system of nodes usually organized by a community of interest (CoI) connected together by a common, secure protocol – usually TCP/IP. Typically, each node executes its own copy of software that securely distributes data between the nodes. Data are generally contained within a transaction each node receives, interprets, and processes independently of the other nodes. Data can be as simple as base types such as integer, double, or string, or as complex as an entire document. All the data required to process the transaction may or may not be contained within the transaction or even the DIDO Network.

In most networks there is a one-to-one relationship between the system of nodes and fungible data managed by the network. For example, within the Bitcoin network, the *coin* is the Bitcoin. The system is solely dedicated to managing Bitcoin. In contrast, although Ether is the basic coin that drives the community and the system of nodes, Ethereum's software allows for multiple kinds of fungible data to be maintained within one network. For example, a single Ethereum network could manage Ether, customer loyalty reward points, and a registry of births and deaths.

Sometimes there is a need for multiple networks, each comprised of a different system of nodes. For example, there may be public networks or private (restricted) networks. Each network is centered on a particular kind of fungible data. Alternatively, the network might be aggregated into broader classifications such as those that have to do with currency (Ether, Bitcoin, etc.) and those that are based around customer loyalty reward programs (e.g., frequent flyer, guest, buyer programs). Still other networks might be based on public records (e.g., births, deaths, divorces), or commodities such as grains or metals. Governments or large corporations may decide to create private networks since the base of nodes they own and control is large enough for redundancy and security and can run on a private intranet.

> **NOTE:** Refer to 2.1.7 Relevant Community Standards for this view.

2.1.3 Ecosystem View

return to Stakeholder Views

The perspectives of Domain and Ecosystem Communities of Interest (CoIs) are different. Whereas a Domain CoI focuses on a specific, real world solution or implementation to a specific concrete problem (i.e., a thing), an Ecosystem CoI focuses on the interactions and links between subordinate Ecosystems (i.e., sub-Ecosystems) and Domain CoIs. Governance of the overall enterprise Data Model (DM) and individual datastores for the enterprise should remain unchanged from the pre-DIDO or non-DIDO state; it is still hierarchical in nature with a definite chain of command and specific responsibility attributed to individuals within the organization. The primary focus of the DIDO Ecosystem is coordination of various other CoIs. For example, one Ecosystem CoI could coordinate other sub-Ecosystem and Domain CoIs to ensure consistent perspective and context across for interoperability across all the CoIs.

A well run DIDO CoI should have a charter that documents:

- the rules of engagement between DIDO CoI members
- the approval processes for changes to the charter and for releases, such as software and configuration
- trouble report procedures (capturing, distributing, resolving, disseminating afterwards)
- maintenance procedures (requesting, frequency, dissemination of maintenance reports)

Ideally, a DIDO CoI should have a chairperson and a board of directors. Some DIDO CoIs may add a technical board or architectural board to oversee technical changes in the product and dedicate the board of directors to oversee governance of the DIDO CoI.

The following diagram represents the various kinds of DIDO CoIs and their relationship to each other.

- DIDO ecosystem is comprised of 1 or more DIDO platforms
- DIDO ecosystem is comprised of 1 or more DIDO domains
- DIDO domain community is comprised of of 1 or more immutable data objects
- DIDO exchange can bridge 1 or more immutable data objects within a DIDO community
- DIDO exchange can bridge 1 or more DIDO communities
- DIDO exchange can bridge 1 or more DIDO ecosystems

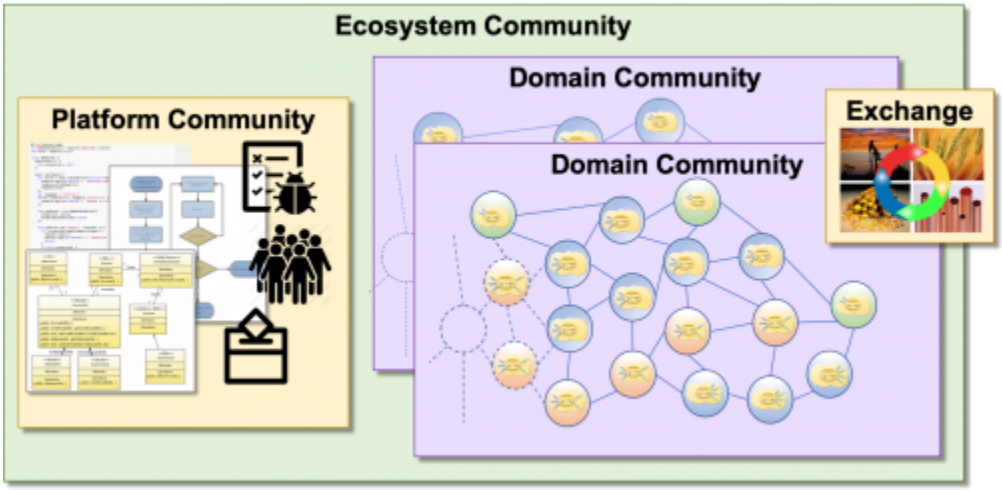


Figure 7: DIDO Ecosystem

NOTE: Refer to 2.1.7 Relevant Community Standards for this view.

2.1.4 Ecosphere View

return to Stakeholder Views

The Ecosphere View comprises the set of all known ecosystems within the ecosphere and the interactions of the ecosystems. Outside of a limited set of use cases, a DIDO cannot function independently, especially when a DIDO is ultimately meant to be part of an enterprise. The description of the DIDO ecosystem works well when everything is defined as a DIDO using greenfield development. However, greenfield development for an enterprise probably will not happen, nor should it happen, given the large amount of legacy data information and processes held outside of the ecosystem. Often, this type of development is referred to as brownfield development. This external data, referred to as *ancillary data*, **is** the immutable data object.

Figure 8: Ecosphere

The ecosphere concept is a way to encapsulate the ecosystem and external ancillary data required to make the individual domains within the ecosystem functional. An individual DIDO domain can access other data sources outside its domain or, for that matter, even within its ecosystem.

> **NOTE:** Refer to 2.1.7 Relevant Community Standards for this view.

2.1.5 Exchange View

return to Stakeholder Views

An exchange is responsible for the exchange of information between immutable data objects. An exchange can be wholly contained within a domain, act as a bridge between domains, or even span across ecosystems or ecospheres. Obviously, the complexity of an exchange increases as the range of what is exchanged moves from just internally within a domain and crosses ecosystem or ecosphere boundaries.

An exchange is a process that exchanges one kind of immutable data object for another. Bitcoin would not be worth much if there were no way to exchange it for existing currencies such as euro, dollar, etc. Exchanges may be completely contained within a DIDO community network or act as the bridge between networks within the community; or be used to transfer immutable data objects between communities (i.e., Bitcoins to Ethereum, Iotas, or commodities such as gold, silver and grains).

An example of an exchange that could operate completely within a community might be for customer rewards programs that include customer loyalty reward programs.

Standards

Technical Standards

- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- OMG: Ontology Definition Metamodel (ODM)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)

de facto Standards

- None at this time

Tools

- None at this time

2.1.6 Enterprise View

return to Stakeholder Views

Referring to the figure in Section 2.1 Stakeholder Views, one can observe that enterprises roughly map to an ecosphere; however, the enterprise does not necessarily control the ecosystems, platforms, or domains it now encompasses. From a DIDO perspective, all the ancillary data (i.e., external data) is accessed through the use of an *oracle*. There can be any number of oracles available to the node within a domain; these oracles can access data from centralized, decentralized, or other DIDOs. Some examples of oracles[14)]:

- Another domain within the same ecosystem
- Documents
- Relational databases
- NoSQL databases
- Flat files
- Web services
- Application services
- Event logs
- Retail transactions
- Bank account records
- Stock market streams

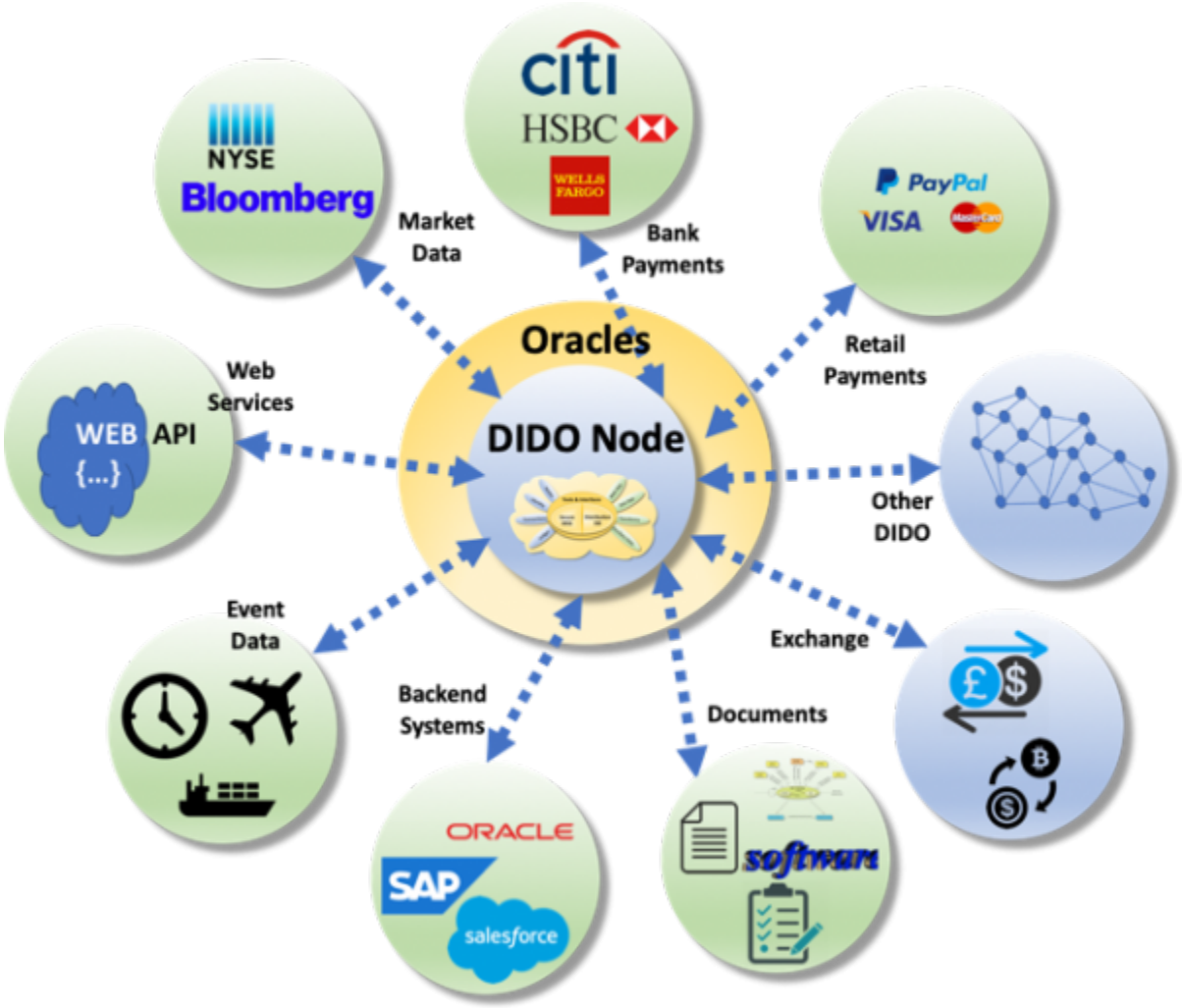


Figure 9: Oracles

One very important aspect of an enterprise with DIDO implementations is the heavy reliance on computer networks. These networks span all kinds of devices and operate in all kinds of environments, many of which rely on Disadvantaged Intermittent Links (DILs). Some examples are: nodes on mobile devices such as cell phones; submarines that are out of contact for long periods of time; nodes hit by natural and/or man-made disasters such as storms, earthquakes, or fire; devices that sleep to save power.

14)

This term should not be confused with the DBMS product from the Oracle Corporation; it is merely a term referring to something that is a good source of information.

2.1.7 Relevant Community Standards

return to Stakeholder Views

The following standards are applicable to all stakeholder views.

Technical Standards

- RFC2026 - The Internet Standards Process
- ISO 9001:2015 Quality management
- ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
- ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE
- ISO/IEC 25001:2014 SQuaRE -- Planning and Management
- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- ISO/IEC 25012:2008 SQuaRE -- Data Quality Model
- ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide
- ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements
- ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality
- ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality
- ISO/IEC 25030:2007 SQuaRE -- Quality Requirements
- ISO/IEC 25040:2011 SQuaRE -- Evaluation Process
- ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators
- ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability
- ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes

de facto Standards

- TODO: How to create an open source program
- TODO: Measuring your open source program's success
- TODO: Tools for managing open source programs
- TODO: Using open source code
- TODO: Participating in open source communities
- TODO: Recruiting open source developers
- TODO: Starting an open source project
- TODO: Improve your open source development impact
- TODO: Shutting down an open source project
- TODO: Building leadership in an open source community
- TODO: Setting an Open Source Strategy

2.2 Technical Views

return to Technical Views

Technical Views are intended to define at a high level the various components of larger DIDO systems. They outline generic parts, subparts, and interconnections of those parts and subparts for a DIDO system. For each part or subpart, a list of potential standards, best practices, and tools is provided. There are multiple views, subdivided into two main areas:

2.2.1 Fundamental Views

return to Technical Views

Fundamental Views are sets of standards that apply throughout all DIDO Ecospheres and are not tied to any specific DIDO component; they relate simply to good system and software engineering. These fundamental views are geared not just to software, but refer to other aspects of a DIDO lifecycle such as requirements, software quality, system and software assurance, and the specification of interfaces to the outside world.

There is no intended hierarchy in the following list of fundamental views.

2.2.1.1 System of Systems (SoS)

return to Fundamental Views

To go beyond simply filling the roles traditionally fulfilled by cash, DIDOs (e.g., cryptocurrencies) need to become part of a fully distributed System-of-Systems (SoS) rather than a single monolithic product offering from a single source. Although the use of OSS helps mitigate the multiple source issue, it is not the panacea some envision because the specification of the system is by definition the current source code of the OSS.[15)] The migration from a single monolithic product offering to an SoS means there should be multiple implementations available for most of the systems (or components) that comprise the DIDO Ecosystem (or Universe). Multiple DIDO implementations greatly reduce the risk of any part of the system being compromised by a single system, subsystem, or component failure, resulting in an even more robust network.

Furthermore, each component must be deterministic in its behavior, meaning that given the same set of inputs, the outputs will always be the same. In other words, not only will all the same implementations of a node produce the same output, but multiple implementations of a component within a node will provide the same results given the same inputs. In OSS systems, the OSS implementation **is** the reference implementation and provides the baseline definition of behavior resulting in a set of specific outputs for specific inputs. Selecting a component should be left to the individual participants in the network and treated as a business decision based on trust, mutual interests, and history. This means the ideal for all components is to have at least two different implementations, with each implementation acting to provide independent validation and verification of the other. This is not so different from current systems, where each node within the blockchain running the same code and getting the same results validates the transaction; differences in the results indicate a potentially compromised node.

For example, a successful DIDO could have worldwide usage (i.e., Bitcoin, Ethereum, etc.). With an SoS approach, each country in which a transaction is executed can have its own implementations of the various components and these implementations can reflect the rules and regulations on reporting and logging of the governing organization. For example, the Swiss might not want to have their transactions reported to the USA, China, Russia or even EU members owing to their unique privacy laws and Data Residency issues. [16)] They would therefore select components that meet the needs of the Swiss rather than the world at large.

Technical Standards

- OMG: Systems Modeling Language (SysML)
- OMG: Unified Architecture Framework (UAF)
- ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality

de facto Standards

- None at this time

15)

Section 2.2.1.2 covers the Open Source Paradigm in more detail.

16)

Object Management Group (OMG), “Data Residency Challenges and Opportunities for Standardization,” March 2017. [Online]. http://www.omg.org/cgi-bin/doc?mars/17-03-22.pdf.

2.2.1.2 Open Source Paradigm

return to Fundamental Views

Using a DIDO is not just a simple shift in policies, procedures, and practices. It is a change in the entire architectural paradigm. Moving away from centralized control to distributed requires a complete change in how the system is normalized into systems, subsystems, components, etc. It also requires a shift in the basic underlying principles of the system. DIDOs are generally:

- Comprised of thousands if not millions of independent nodes
- Outside the control of any one individual or corporation
- Lacking any centralized authority; decisions are made by consensus

The DIDO architecture does not represent a single unified enterprise, but rather a loosely defined confederation of domains that requires systems integration (SI)[17]. Although SI is not new to enterprises, the granularity and types of components require a rethink. Within the DIDO environment, the definition of a platform shifts from hardware, operating system, software languages, and services (e.g., web, app, database) components to the DIDO Platform components. It is the responsibility of the DIDO Platform to isolate the enterprise from traditional platform concerns.

The granularity of the data elements within an enterprise can also shift to smaller, more isolated objects, which represent only a portion of the traditional Data Model (DM). In other words, the enterprise's data model is not going to be deployed into a single DIDO, nor should it. Enterprise data stores will continue to be needed but will be augmented and complemented by the DIDO. Some data will reside completely within the enterprise data stores, some data will reside completely within the DIDO, and some data will straddle both. Data that straddles both will require the definition of policies and procedures to ensure their data integrity.

Relevant Open Source Standards

The cultural shift from a stove-piped corporate or enterprise culture with almost complete control, to being a systems integrator participating in numerous distributed communities covering a wide range of domains, requires committed leadership and concerted effort by all the players.

Technical Standards

- None at this time.

de facto Standards

- There are none at this time but the Open Source Communities often rely heavily on the products of both Technical Standards Bodies and de facto Standards Bodies in building their projects.

- There are many guides available for participating in Open Source initiatives. **Talk Openly Develop Openly** (TODO) provides an extensive reading list.[18]. TODO also provides the following excellent guide as a place to start: TODO: Participating in open source communities.

[17]

System Integrator - An individual or organization that builds systems from a variety of diverse components. With increasing complexity of technology, more customers want complete solutions to information problems, requiring hardware, software and networking expertise in a multi-vendor environment. https://www.pcmag.com/encyclopedia/term/52450/systems-integrator

[18]

TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

2.2.1.3 Case Management

return to Fundamental Views

A major problem confronting DIDO Communities is the development of customer support to deal with issues encountered with either DIDO Transactions, Smart Contracts (especially those written externally by third parties), or DIDO OSS. For example, people have come to expect that when they use financial services, there will be recourse if transactions have unintended consequences. Recently, Coinbase, which provides an easy-to-use service for trading cryptocurrencies such as Bitcoin, Litecoin, and Ethereum, was hit with class action lawsuits alleging insider trading and also unlawful and unfair business practices.[19)] In traditional banking, lack of recourse was part of the motivation behind the Dodd-Frank Wall Street Reform and Consumer Protection Act [20)] , especially Section 1034 “Response to Consumer Complaints and Inquiries.” [21)] Case Management must be applied to both of the organizational parts of DIDO Communities: software and fungible data (i.e. currency) management.

There are several categories of problems that can arise in a distributed system:

- Those involving distributed data
- Those involving the software used to distribute the data
- Those on a local node (machine)

Problems with Distributed Values (Domain Issues)

Problems that arise on the node network with the values stored on a node or set of nodes are generally domain issues. These cases generally have to do with the implementation of a DIDO Domain (i.e., Bitcoin cryptocurrency versus the Bitcoin platform). Therefore, the case is reported to the DIDO Domain. If the problem can be resolved at the domain level that's as far as the case needs to go. However, sometimes these cases need to be resolved at the domain and platform levels, thereby requiring two cases.

Problems with Distributed Software (Platform Issues)

Problems that arise on the node network having to do with conflicts in valid values stored on a node or set of nodes are generally platform issues. Generally, there should be no conflict with the values on any of the nodes since the DIDO implementations employ consensus methodologies, which form a large part of the added value of the individual DIDO platforms. For example, Bitcoin uses a Proof of Work (POW) methodology, whereas Ethereum and others use a Proof of State (PoS) methodology.

Problems with Node

Sometimes a node within the node network has problems. In a DIDO that has built-in redundancy, validation, and verification, this is generally not a problem and should be handled by the the original

design. However, if nodes with a particular configuration (i.e., hardware, operating system, patches, security software, network cards, etc.) have issues, this could have consequences on the overall health of the domain.

Summary

Regardless of the location of the source for the case, most domains or platforms use Open Source Software and a bug tracking process based on a particular bug tracking tool such as Bugzilla, Jira, or Git-bug.

Generally, when consensus is reached for the correct value and requires a software upgrade, these cases are resolved using a Soft Fork. When consensus cannot be reached, a Hard Fork can occur within the domain.

Standards

Technical Standards

- OMG: Case Management Model and Notation (CMMN)

de facto Standards

- Bitcoin: Bitcoin Improvement Proposals (BIPs)
- Ethereum: Ethereum Improvement Proposals (EIPs)

Tools

- Tools: Bug and Issue Tracking

19)
S. Chang, “Coinbase Hit with 2 Class Action Lawsuits: Accused of Insider Bitcoin Cash Trading,” 4 March 2018. [Online]. https://www.investopedia.com/news/coinbase-hit-2-class-action-lawsuits-accused-insider-bitcoin-cash-trading/

20)
Investopedia, “Dodd-Frank Wall Street Reform and Consumer Protection Act,” http://www.investopedia.com/terms/d/dodd-frank-financial-regulatory-reform-bill.asp.

21)
International Association of Risk and Compliance Professionals (IARCP), “Dodd Frank Act Text Section 1034,” 2010. http://www.dodd-frank-act.us/Dodd_Frank_Act_Text_Section_1034.html.

2.2.1.4 Assurance

return to Fundamental Views

The existing strategy for software and system assurance is already defined by the Systems and software Quality Requirements and Evaluation (SQuaRE). It establishes a common framework for analysis and exchange of information related to system assurance and trustworthiness, and defines the following kinds of assurance that need to be addressed: Information Assurance (IA), Safety Assurance (SfA), Software Assurance (SwA), Mission Assurance (MA) and System Assurance (SysA).

Assurance does not yield binary true / false answers. Assurance is a measure of risk which is a probability or threat of damage, injury, liability, loss, or any other negative occurrence that is caused by external or internal vulnerabilities, and that may be avoided through preemptive action.[22)]. Assurance is best handled using Structured Assurance Case Metamodels (SACMs) for each of the assurances detailed above. A DIDO community of interest (CoI) best interest is to provide assurance measurements of their software, especially those CoIs that are offering “coinage” products to provide formal SACM results.

22)

Business Dictionary, Accessed 1 June 2020, http://www.businessdictionary.com/definition/risk.html

2.2.1.5 Quality

return to Fundamental Views

The ISO/IEC 25010 standard provides consistent terminology for “specifying, measuring and evaluating system and software product quality”.[23] The Consortium for Information & Software Quality (CISQ) [24] provides the following diagram, highlighting eight ISO defined software quality characteristics and their associated sub-characteristics.[25] These ISO defined characteristics must be applied to both DIDO software and coinage organizations (e.g., Bitcoin, Ethereum, IOTA), especially Reliability, Performance Efficiency, Security, and Maintainability, since these are candidates for automation.

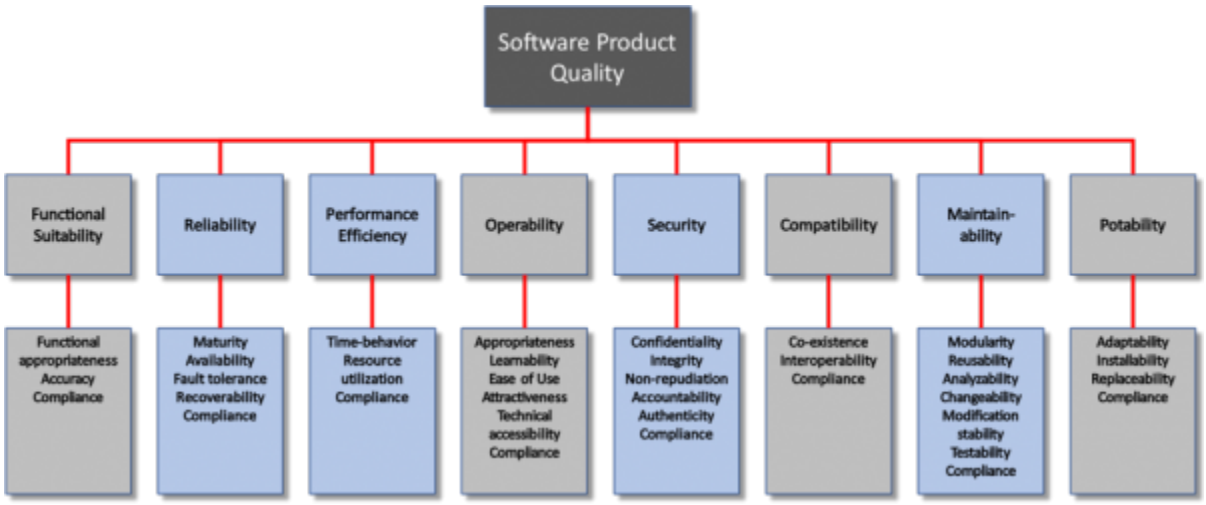


Figure 10: Quality Characteristics and Measures Specifications

Management

Management is part of SQuaRE and defines all common models, terms, and definitions referenced by all other standards from the SQuaRE series.

- ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE
- ISO/IEC 25001:2014 SQuaRE -- Planning and Management
- ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
- ISO 9001:2015 Quality management

Modelling

Modelling is part of SQuaRE and provides detailed quality models for computer systems and software products, quality in use, and data.

- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- ISO/IEC 25012:2008 SQuaRE -- Data Quality Model

Measurement

Measurement is part of SQuaRE and includes a software product quality measurement reference model, mathematical definitions of quality measures, and practical guidance for their application.

- ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide
- ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements
- ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality
- ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality

Requirements

Requirements are part of SQuaRE and help specify quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed, or as input for an evaluation process.

- ISO/IEC 25030:2007 SQuaRE -- Quality Requirements

Evaluation

Evaluation is part of SQuaRE and provides requirements, recommendations, and guidelines for software product evaluation.

- ISO/IEC 25040:2011 SQuaRE -- Evaluation Process
- ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators
- ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability

23)

International Association of Risk and Compliance Professionals (IARCP), “Dodd Frank Act Text Section 1034,” 2010. http://www.dodd-frank-act.us/Dodd_Frank_Act_Text_Section_1034.html.

24)

International Standards Organization (ISO), “The ISO/IEC 25000 series of standards,” http://iso25000.com/index.php/en/iso-25000-standards?limit=4&start=4.

25)

Consortium for Information & Software Quality (CISQ) http://it-cisq.org/.

2.2.1.6 Interfaces

return to Fundamental Views

Interfaces exist between the various independent components of the DIDO and are therefore fundamental to the integration of DIDO components into a system. Bidirectional interfaces are defined between:

- hardware-to-hardware
- hardware-to-software
- hardware-to-human
- software-to-human

Hardware

Hardware interfaces are how the outside world connects to physical devices such as disk drives, monitors, keyboards, internal buses, batteries, internet ports, cameras, microphones, and sensors. These interfaces are described by the mechanical, electrical, and logical signals at the interface and the protocol for sequencing them.[26)] Fortunately, most of these interfaces are integrated into computer nodes and do not need to be addressed here. For example, the data from the sensor can be considered an Immutable Data Object and thus sent through the node network, but the actual acquisition of that data is a detail for nodes to implement.

Refer to Section 2.2.1.6.1 Platform Interface for more detail on Hardware interfaces.

Software

Software interfaces isolate the underlying complexity of functionality provided by a software product or package through the use of an Application Programming Interface (API). The end result of this is to treat software products or packages as opaque blackboxes.

Refer to Section 2.2.1.6.2 Software Interfaces for more details on this subject.

Human

Human interfaces within DIDOs are bidirectional and primarily cover the software-to-human components. A software Application Programming Interface (API), although defining interactions between people and software, is classified as a software interface and not a human interface. Human interfaces cover the human interaction between the humans and an operational system; examples include mouse, keyboard, trackpad, and windows presented to users of the system.

Refer to Section 2.2.1.6.3 Human Interfaces for more details on this subject.

26)

Govindarajalu, B. (2008). “3.15 Peripheral Interfaces and Controllers - OG”. IBM PC And Clones: Hardware, Troubleshooting And Maintenance. Tata McGraw-Hill Publishing Co. Ltd. pp. 142–144. ISBN 9780070483118. Retrieved 15 June 2018.

2.2.1.6.1 Platform Interface

return to Interfaces

A **Platform** is the computing environment that software resides-in and executes-on. In the DIDO context, the software is the Distributed Application (ĐApp or DApp) that runs on a particular DIDO Platform (i.e., Bitcoin, Ethereum, IOTA, IPFS, DDS, etc). The DIDO Platform is comprised of the following abstractions:

1. the computer hardware (real or virtual)
2. the operating system (OS)
3. other layers of software used to support the application (DIDO Platform)

The first layer, the Operating System, abstracts access to the actual hardware such as memory, disks, networks etc. by providing Application Programming Interfaces (APIs) for applications to use. In other words, the application is tied to the OS rather than a particular hardware configuration. However, this still ties an application to an OS.

The second layer of abstraction occurs when the interface to the OS becomes standardized, as it is with the IEEE Portable Operating System Interface (POSIX) 1003.1-2016[27]. The POSIX standard supports applications portability at the source code level and includes provisions for a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs.[28]

The third layer (DIDO Platform) provides an API for a Distributed Application (DAPP) to reside-in and execute-on. Access to the other DIDO components is tightly controlled for security reasons. Unfortunately at this time, there is no standardized abstraction for DIDO Platforms.

Note: A DIDO Node within the DIDO Network provides a platform or subsetted platform so that DApps can be run on each node.

Standards

Technical Standards

- None at this time

de facto Standards

- Bitcoin: Developer's Guidance
- Bitcoin: Bitcoinj Developer's Documentation
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project

- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project

Tools

- None at this time

27)

IEEE Std 1003.1, 2016 Edition, IEEE Standard for Information Technology - Portable Operating System Interface (POSIX), includes IEEE Std 1003.1-2008, IEEE Std 1003.1-2008/Cor 1-2013, IEEE Std 1003.1-2008/Cor 2-2016.

28)

POSIX Product Standards, 1003.1-2016 Base, http://get.posixcertified.ieee.org/docs/base-2016.html

2.2.1.6.2 Software Interfaces

return to Interfaces

Defining formal, well-formed, consistent, and uniform interfaces between nodes within the network and the interfaces between the components within a node is essential for providing stable products that can evolve over time. For example, the distributed nature of a DIDO means the system must be designed to account for inherent mismatches between versions of software running on each node within the network, especially since it takes time for updates to flow through the system (sometimes referred to as “reboot the world problem”).

Software interfaces are the most sensitive to component changes within the DIDO. The problem is similar to the logging problem, requiring cross platform, cross programming language requirements, and cross version support; not everything can be written in Java, Python, or even PHP.

Some other tools that might be useful in a distributed environment are debuggers, network analyzers, and discovery aids. Furthermore, given that the system supports all kinds of operating environments and multiple implementations, clear, unambiguous definitions of software interfaces are essential.

Formally defined and maintained Application Programming Interfaces (APIs) are necessary for all the DIDO parts to work together seamlessly.

Standards

Technical Standards

- OMG: Interface Definition Language (IDL)

de facto Standards

- None at this time

Tools

- None at this time

2.2.1.6.3 Human Interfaces

return to Interfaces

Until humans evolve to have one culture and speak with one language, there is no standard way to interface with humans. In many ways human interfaces are cultural and are dependent on the human experience. Human interfaces have evolved, and are evolving, so that the effectiveness of the interface is in many ways fickle, trendy, and highly subject to bias. However, this does not mean that there are no guiding principles for human interfaces.

Projects should have a Graphical User Interface (GUI) Style Guide and projects should remain consistent within those guidelines.

Note: Because of the cultural nature of user Interfaces, only *guidance* is listed, not standards.

Standards

Technical Guidance

- ADA Website Accessibility Under Title II of the Americans with Disability Act (ADA) - ADA Best Practices Tool Kit for State and Local Governments, Website Accessibility Under Title II of the ADA, Chapter 5, https://www.ada.gov/pcatoolkit/chap5toolkit.htm
- Accessibility of State and Local Government Websites to People with Disabilities, U.S. Department of Justice, Civil Rights Division, Disability Rights Section https://www.ada.gov/websites2.htm

de facto Guidance

Every project and program usually develops its own guidelines for user interfaces (UI). Often, these are modeled after some of the guidance offered by major corporations known for having “good” products.

- Android Developer, User Interface and Design, https://developer.android.com/guide/topics/ui
- Apple, Mac OS Developer, https://developer.apple.com/design/human-interface-guidelines/macos/overview/visual-index/
- Apple IOS Developer, https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/
- GNOME Human Interface Guidelines, https://developer.gnome.org/hig/stable/
- KDE Human Interface Guidelines, https://hig.kde.org/
- Microsoft Designing a User Interface, https://docs.microsoft.com/en-us/windows/win32/appuistart/designing-a-user-interface

Tools

There are many tools and frameworks available. The best advice is to remember that DIDO deployments are not as agile as stand-alone applications and require significant effort to keep the distributed applications in sync. Choose tools and frameworks keeping that in mind.

2.2.1.7 Tools

return to Fundamental Views

Tools are used to create, debug, maintain, or otherwise support other programs, applications, and communities. Within the DIDO architecture the application covers the DIDO platform, domain, exchange, node, and node network.

2.2.1.7.1 Logging

return to Tools

One of the most critical requirements for distributed systems is the need to support common logging of events, information, and errors as well as support a multitude of programming environments (e.g., Java, C/C++, C#, JavaScript, Windows, UNIX, Linux).

> *Data logging is the process of collecting and storing data over a period of time in order to analyze specific trends or record the data-based events/actions of a system, network or IT environment. It enables the tracking of all interactions through which data, files or applications are stored, accessed or modified on a storage device or application.* [29)]

The protocol used for logging needs to be a redundant parallel system that reduces the dependency on the same infrastructure as the nodes and the node network to communicate. For example, if the node and node network use the same messaging software as the logging system, when a problem occurs, critical logging may also be down.

Standards

Technical Standards

- RFC5424 - The Syslog Protocol (SYSLOG)

de facto Standards

- Oracle: Java logger API
- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala

Tools

- Tools: Logging Tools

29)

Techopedia, “Techopedia Data Logging,” 2 December 2017. https://www.techopedia.com/definition/596/data-logging

2.2.1.7.2 Semantic Web

return to Tools

The Semantic Web extends the World Wide Web (www) through standards developed by the World Wide Web Consortium (W3C).[30] These standards provide common data formats and exchange protocols on the web, typically using Resource Description Framework (RDF). RDF facilitates data merging even when underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all data consumers to be updated or modified.[31] "The Semantic Web provides a common framework allowing data sharing and reuse across applications, enterprises, and community boundaries".[32] The Semantic Web is therefore regarded as an integrator across different content, information applications, and systems.

Standards

Technical Standards

- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- OMG: Ontology Definition Metamodel (ODM)

de facto Standards

- None at this time

Tools

- None at this time

30)
"XML and Semantic Web W3C Standards Timeline" (PDF). 2012-02-04. http://www.dblab.ntua.gr/~bikakis/XML%20and%20Semantic%20Web%20W3C%20Standards%20Timeline-History.pdf

31)
Resource Description Framework, https://www.w3.org/RDF/

32)
"W3C Semantic Web Activity". World Wide Web Consortium (W3C). November 7, 2011. http://www.w3.org/2001/sw/

2.2.1.7.3 Open Source Communities

return to Tools

An important tool used within DIDOs is the use of Open Source Software (OSS). The use of OSS is not simply publishing software and allowing people to use it, but requires the establishment of an entire community with carefully developed rules and procedures to guide the design, development, release, maintenance, and sunsetting of the software.

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- Community tools

2.2.2 Node Network View

return to Technical Views

There are slightly different variations for the definitions of a Node Network, Node, Full Node, Light Node, and Miner Node depending on the DIDO Platform.[33),34),35),36)]

- The **Node Network** is a collection of interconnected computers communicating over a network of equally privileged computers (i.e., there are no central authoritative computers). The node network is sometimes referred to as a **peer-to-peer (P2P) network**.
- A **Node** is an individual computer that participates as an equal within the node network. A node is sometimes referred to as a **peer**. Nodes receive input, perform one or more operations on those inputs, and returns an output.

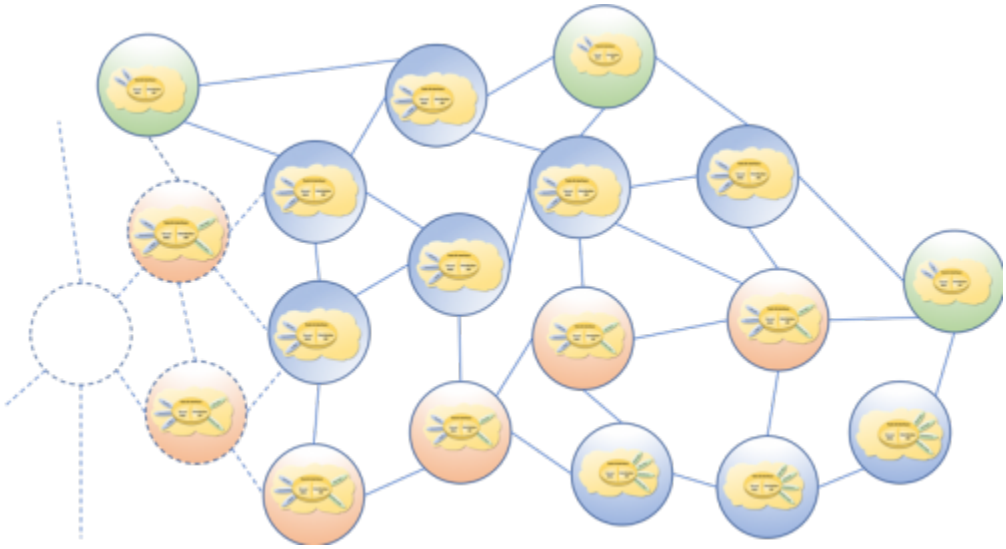


Figure 11: Node Network of Nodes.

33)

"What is a Bitcoin node?", Nate Eldredge, 14, December 2013, https://bitcoin.stackexchange.com/questions/18736/what-is-a-bitcoin-node

34)

"What are Ethereum Nodes And Sharding?", Ameer Rosic, 2017, https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/

35)

"IOTA Full Node—That's Why a Full Node Is Important for IOTA!", Marko Vidrih, February 2019, https://medium.com/altcoin-magazine/iota-full-node-thats-why-a-full-node-is-important-for-iota-1c46280d7712

36)

“Introduction to IPFS: Run Nodes on Your Network, with HTTP Gateways”, Ross Bulat, 3 December 2018, https://medium.com/@rossbulat/introduction-to-ipfs-set-up-nodes-on-your-network-with-http-gateways-10e21ea689a4

2.2.2.1 Network View

return to Node Network View

The Network View looks at the DIDO as a single entity. Even though the network is comprised of a collection of individual nodes, individual nodes work as one because they:

- Interact as a peer in community of peers (i.e., Peer to Peer (P2P))
- Use a single Data Object (i.e., base coinage)
- Process transactions according to the rules of the community

For example, there is a system of nodes involved in the lifecycle of Bitcoins. However, it makes no sense for public records such as births, deaths, marriages, and divorces to be in the Bitcoin system of nodes. Thus, the public records could form their own system of nodes using the Bitcoin software but restrict this network to storage of pubic records.

Obviously, Bitcoin's main purpose is to transfer assets around the world at high speed and with low overhead. However, these are not the capabilities motivating the use of DIDO networks for low volatility public records such as those which reflect the natural rates of births, deaths, marriages, and divorces. Such applications are usually under the jurisdiction of a single country or countries that have reciprocity agreements or treaties.

Similarly, the need to establish a private system of nodes might exist for internal use only users (e.g., government enclaves, large corporations) that have enough distributed resources to support the network. Although the current cryptographic protocols provide "security" to a DIDO, with the advent of quantum computing [37], a reliance on these algorithms in the future for highly classified or private data may not be acceptable. These risks provide even more justification for the development of private DIDO networks.

As a general rule, the larger the system of nodes, the more secure and tamper-proof the data held within the network becomes, which means that for networks to be viable from a security perspective, the number of nodes in the collection might have a minimum.

- 2.2.2.1.1 Secure Messaging
- 2.2.2.1.2 Transport
- 2.2.2.1.3 Security
- 2.2.2.1.4 Protocol
- 2.2.2.1.5 Distribution Software

37)
MIT Technology Review, "Quantum Computers Pose Imminent Threat to Bitcoin Security," 8 November 2017.
https://www.technologyreview.com/s/609408/quantum-computers-pose-imminent-threat-to-bitcoin-security/

2.2.2.1.1 Secure Messaging

return to Network View

The lowest functional layer of a node component is secure messaging. It is essential that a distributed system be able to have trusted, reliable, tamper resistant, and anti-snoop data flow between nodes in the network. Secure messaging must also include a standardized, consistent, and predictable way to marshal data between nodes. For example, the Endianness of each node may be different. This requires a solid transport mechanism, an excellent security infrastructure, and a standardized way to distribute data quickly and efficiently.

Standards

Technical Standards

- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC7061 - eXtensible Access Control Markup Language (XACML) XML Media Type
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- OMG: Data Distribution Service (DDS)
- OMG: DDS Security (DDS-SECURITY)

de facto Standards

- ZeroMQ Distributed Messaging
- Google: gRPC
- Google: Protocol Buffers
- Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)
- Linux Foundation: Open Messaging
- BIP 0070 - Payment Protocol

2.2.2.1.2 Transport

return to Network View

A cornerstone of DIDO networks is the transport component, enabling nodes to communicate with a very high degree of trust and reliability. Although it is possible to create a network of nodes that do not use Internet Protocol (IP), the breadth and acceptance of such an endeavor is questionable.

> *DIDO relies on the transport layer as defined by the Open Systems Interconnection (OSI) model. The transport layer is the layer in the open system interconnection (OSI) model responsible for end-to-end communication over a network. It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components. The transport layer is also responsible for the management of error correction, providing quality and reliability to the end user. This layer enables the host to send and receive error corrected data, packets or messages over a network and is the network component that allows multiplexing.*[38)].

Standards

Technical Standards

- RFC0147 - The Definition of a Socket
- RFC0791 - Internet Protocol (IPv4)
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)

de facto Standards

- Google: Protocol Buffers
- ZeroMQ Message Transport Protocol (ZMTP)

Tools

- None at this time

38)

Techopedia, “Transport Layer,” 30 November 2017.https://www.techopedia.com/definition/9760/transport-layer

2.2.2.1.3 Security

return to Network View

A DIDO is focused on two elements of security:

Internet Security: intended to protect the data flowing through the network. A major part of the functionality of a DIDO is its distributed nature over the internet and trust that it processes valid and accurate transactions.

> *Internet Security is essential in establishing trust to its success. Internet security is a catch-all term for a very broad issue covering security for transactions made over the Internet. Generally, Internet security encompasses browser security, the security of data entered through a Web form, and overall authentication and protection of data sent via Internet Protocol.*[39)]

Information Security: intended to protect the confidentiality, integrity, and availability of the information contained within the DIDO network.

> *Information security (IS) is designed to protect the confidentiality, integrity and availability of computer system data from those with malicious intentions. Confidentiality, integrity and availability are sometimes referred to as the CIA Triad of information security. This triad has evolved into what is commonly termed the Parkerian hexad, which includes confidentiality, possession (or control), integrity, authenticity, availability and utility.*[40)]

Standards

Technical Standards

* OMG: DDS Security (DDS-SECURITY)

de facto Standards

* None at this time

Tools

* None at this time

39)

Techopedia, “Techopedia Internet Security,” 30 November 2017. https://www.techopedia.com/definition/23548/internet-security
40)

Techopedia, “Information Security (IS),” 30 November 2017. https://www.techopedia.com/definition/10282/information-security-is.

2.2.2.1.4 Protocol

return to Network View

At the most elementary level, a communication protocol is comprised of a set of rules and guidelines that allow two or more nodes to communicate successfully over a network. DIDO networks must have a standardized robust protocol as the basis for processing transactions and to provide other communications between nodes.

> *A protocol is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more computers. Networks have to follow these rules to successfully transmit data.*[41)]

Standards

Technical Standards

- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC0768 - User Datagram Protocol (UDP)
- RFC1112 - Host Extensions for IP Multicasting
- RFC3339 - Date and Time on the Internet: Timestamps
- RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format
- OMG: Data Distribution Service (DDS)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Web-Enabled DDS (DDS-WEB)

de facto Standards

- Bitcoin: Bitcoinj Developer's Documentation
- EIP 1474: Remote Procedure Call (RPC) specification (DRAFT)
- EIP 234: `blockHash` to JSON-RPC filter options (DRAFT)
- EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)
- EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)
- EIP 1193: Ethereum Provider JavaScript API (DRAFT)
- Ethereum: cpp Project

- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- Google: gRPC

Tools

- None at this time

41)

Techopedia, “Technopedia Protocol,” 30 November 2017. https://www.techopedia.com/definition/4528/protocol.

2.2.2.1.5 Distribution Software

return to Network View

Each node has a distribution software component. It is responsible for distributing and coordinating data and software throughout a DIDO network. Probably one of the best-known implementations of distribution software is the Domain Name System (DNS), which forms the backbone of the Internet:

> *Domain name system (DNS) is a hierarchical naming system built on a distributed database. This system transforms domain names to IP addresses and makes it possible to assign domain names to groups of Internet resources and users, regardless of the entities' physical location.*[42)]

However, with the advent of the Blockchain papers by Satoshi Nakamoto [43)] another major player, Bitcoin, has emerged in the distribution software space.

Standards

Technical Standards

- RFC1034 - Domain Names - Concepts and Facilities
- RFC1035 - Domain Names - Implementation and Specification
- RFC3596 - DNS Extension to support IP Version 6
- RFC5011 - Automated Updates of DNS Security (DNSSEC) Trust Anchors
- RFC6376 - DomainKeys Identified Mail (DKIM) Signatures
- RFC6891 - Extension Mechanisms for DNS (EDNS(0))

de facto Standards

- None at this time

Tools

- None at this time

42)
Techopedia, “Techopedia Domain Name Service (DNS),” 30 November 2017. https://www.techopedia.com/definition/24201/domain-name-system-dns.

43)
S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. https://bitcoin.org/bitcoin.pdf.

2.2.2.2 Node View

return to Node Network View

The Node View represents the internals of any particular node within the node network. As represented in the figure below, it is comprised of five layers; however, the layers are not set in stone – each implementation of a node within a domain may be organized differently.

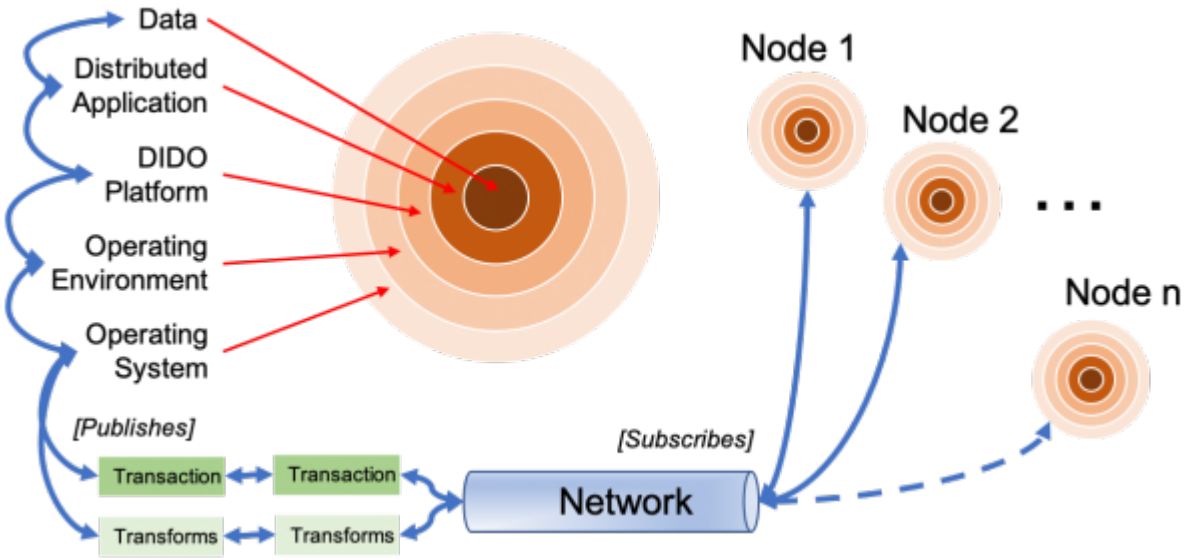


Figure 12: The idealized five layers of a Node.

At the boundary between each layer there is an interstitial layer formed by an Application Programming Interface (API), which is usually defined by one or more technical or de facto standards.

As each transaction, transform, or stream of data flows to or from a node over the network, it must travel through and/or interact with the:

- 2.2.2.2.1 Operating System (OS)
- 2.2.2.2.2 Operating Environment
- 2.2.2.2.3 DIDO Platform
- 2.2.2.2.4 Distributed Applications

2.2.2.2.1 Operating System (OS)

return to Node View

An operating system (OS), in its most general sense, is software that allows a user to run other applications in a computing device, as well as Virtual Machine applications, which emulate another computer. While it is possible for a software application to interface directly with hardware, it is not advisable from a portability or lifecycle perspective. Software applications that access hardware resources or other computer components directly pose a security risk.

Operating systems provide a common, well documented, and tested set of libraries, which abstract the idiosyncrasies of the host computer away from its applications.

An OS primarily manages a computer's hardware resources, including:

- Input devices such as a keyboard, mouse, track pad, touch screens, camera, microphone, scanners, or sensors
- Output devices such as display monitors, speakers, printers, or faxes
- Network devices such as modems, router, wired and wireless Internet Protocol network connections, and Bluetooth
- Storage devices such as internal and external disks
- Memory devices

The OS also manages a computer's:

- CPU
- Processes
- Privileges
- Cache
- Energy, i.e., power management

Standards

Technical Standards

- IEEE 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications (**NOTE:** See UNIX)
- ISO/IEC 23360-1:2006 Linux Standard Base (LSB) core specification 3.1 -- Part 1: Generic specification
- ISO/IEC The Linux Standard Base 5 Specification Series (LSB 5)

de facto Standards

- Apple: Darwin

- Apple: iOS
- Apple: MacOS
- Google: Android
- Microsoft: Windows API

Tools

- None at this time

2.2.2.2.2 Operating Environment

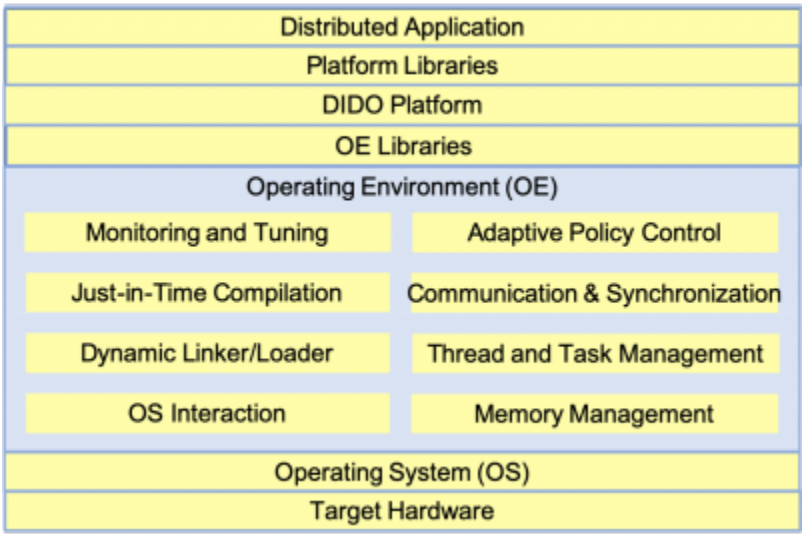
return to Node View

The concept of an Operating Environment (Run Time System) is generally traced back to *Ada*, which defined: an abstract interface to the underlying operating system, thread and task management, as well as mechanisms for monitoring, tuning, and performing dynamic memory management (all to protect against access to unallocated memory, buffer overflow errors, range violations, off-by-one errors, array access errors, and other detectable poor coding practices)[44].

Another major advancement came with Java and the Java Virtual Machine (JVM), which added a Just-In-Time (JIT) compiler, virtual processor, and an interpreter[45].

Microsoft later introduced the .NET framework, which runs on the Windows operating system. .Net includes Web Services, Web Forms, and Windows Forms in the Operating Environment[46]. Subsequently, Mono was introduced and is now sponsored by Microsoft in order to allow cross-operating system development and deployment of .Net applications. The Mono Framework is based on ECMA Standards for C# and the Common Language Runtime[47];

Figure 13: Generalized Operating Environment (OE) components[48]



Standards

Technical Standards

- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from

Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Solidity Language Specification

- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- None at this time

44)

https://en.wikipedia.org/wiki/Ada_(programming_language

45)

“Understanding the JVM Architecture”, Joydip Kanjilal, Developer.com, 16 February 2015, https://www.developer.com/java/data/understanding-the-jvm-architecture.html

46)

“Components of .Net Framework”, DeveloperIn.Net, Jayanthan JVP, http://www.developerin.net/a/39-Intro-to-.Net-FrameWork/23-Components-of-.Net-Framework

47)

“Cross platform, open source .Net Framework”, The Mono Project, https://www.mono-project.com/

48)

“Language Run-Time Systems: An Overview”, Evgenij Belikov, Heriot-Watt University, School of Mathematical and Computer Sciences Riccarton, EH14 4AS, Edinburgh, Scotland, UK https://pdfs.semanticscholar.org/b20c/0295a0661a4c175fcd5acc0ea49e9caf3ca1.pdf

2.2.2.2.3 DIDO Platform

return to Node View

A DIDO platform definition encapsulates the complete environment that supports a running DIDO. The DIDO platform covers the hardware, operating system (OS), DIDO software, and the connection to the network. There are potentially many related DIDO platforms defined for any particular domain. Each DIDO platform can vary the hardware, OS, type of network connection, and potentially the DIDO software if the software can interoperate.

For example, a domain defined for a supply chain might have a set of DIDO platforms defined. A DIDO platform for Windows, Linux, UNIX, Android, Mac OS, and IOS can work on TCP/IP machines or TCP/UDP protocols.

Standards

Technical Standards

- None at this time

de facto Standards

- Bitcoin: Bitcoinj Developer's Documentation
- Bitcoin: Developer's Guidance
- Bitcoin: Bitcoin Improvement Proposals (BIPs)
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- Google: Protocol Buffers

Tools

- None at this time

2.2.2.2.4 Distributed Applications

return to Node View

A **distributed application** is software that does not reside on a single computer, centralized server, decentralized server, or set of clustered servers. The application is distributed among nodes on the network. Each instance of the application on each node executes the same code and gets the same results based on the current state of the data and the input data sent to it. In other words, the distributed application's nodes are deterministic in nature.

Standards

Technical Standards

- RFC6455 - The WebSocket Protocol
- RFC0793 - Transmission Control Protocol
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC1112 - Host Extensions for IP Multicasting
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC0768 - User Datagram Protocol (UDP)
- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification

- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Solidity Language Specification
- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- Tools: Network Traffic Analysis

2.2.2.3 Node Architecture

return to Node Network View

The DIDO node architecture establishes a component model as a reference for evaluating the functionality or data available within a DIDO implementation. The reference components are conceptual in nature; thus they may or may not represent actual components within a DIDO implementation. However, the functionality of these reference components should be part of the implementations.

At the most elementary level, reference components provide a common vocabulary for DIDO implementations. For example, Bitcoin does not have a separate standalone component that performs secure messaging; however, the Bitcoin software *does* provide secure messaging functionality using a protocol built upon cryptography. This binds the secure messaging and protocol together. It is possible to write applications that adhere to this protocol using the cryptographic rules and standards defined by Bitcoin and not use the Bitcoin software; however, the usual approach is to build upon the open source Bitcoin software and use it “as-is.” Another example is where Bitcoin provides an actual component referred to as a Wallet, bearing in mind there are numerous other wallets available that can contain and manage Bitcoins.[49)]

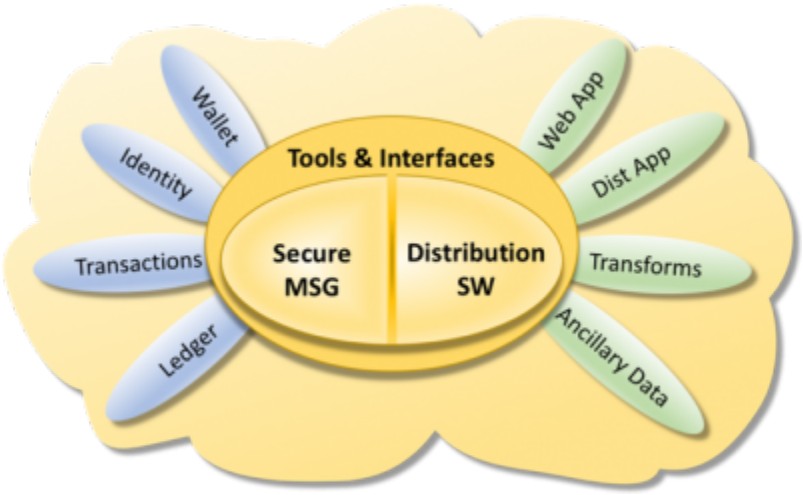


Figure 14: The DIDO Node Component Model

The DIDO node component model presented in this diagram (Figure 14) may look like a traditional stack or layered architecture as described in centralized or decentralized models; however, a very important difference is that the components or a subset of the components of this architecture are usually repeated at every DIDO node that participates in the DIDO network. At a minimum, the secure message and the distribution software components must exist at each node in order for the network to remain distributed and operational. In addition to providing for the core components required to securely communicate, a DIDO node can take on different roles or functions (refer to Figure 15). Some nodes may use all the components within the DIDO node component model while others may only use a subset of these components. For example, a smart contract node may use the identity, transaction, ledger, distributed app, and ancillary data components. In contrast, a Wallet node may just use wallet and identity components.

Figure 15: Examples of kinds of DIDO Nodes

When the entire DIDO network is assembled, it may look something like Figure 16. Each node has the basic common core components and the set of components required for it to fulfill its specific responsibilities.

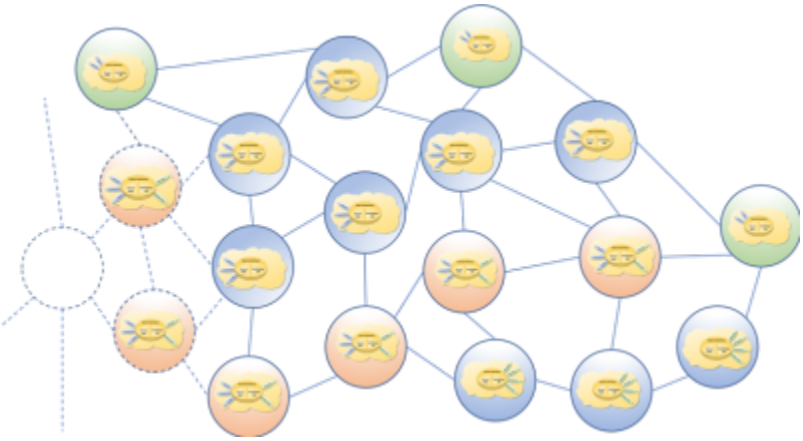


Figure 16: The DIDO Network Model

The DIDO network acts as a single system or entity, in which each node performs the operations required of it as a participant. For example, some DIDO networks may only provide a ledger, account numbers, and support transactions on the ledger. These would require each node to have the secure message, distribution software, ledger, identity and transaction components in common.

All the components within the network must be interoperable at the core functional level. For example, most of the nodes might run the Common Core version 1, whereas a subset might be running version 1.1. As long as the nodes within the network can interoperate and function together, the network is considered viable as a whole.

The interoperability of nodes within the DIDO and overall DIDO network viability become key benefits of DIDO implementations. This means that anything affecting interoperability is critical, and requires that interfaces and interactions (i.e., protocols) between the nodes must be the most conservative of all components. In other words, interfaces and interactions of the nodes are the mission critical aspects of the network and, therefore must be stable from the onset. Interruption or discontinuity in the critical path could result in a fork of the underlying ledger. An example of a fork is the "hard fork" in Ethereum called Byzantium:

> *The Byzantium hard fork is an update to ethereum's blockchain that was implemented in October 2017 at block 4,370,000. It consisted of nine Ethereum Improvement Protocols (EIPs) designed to improve ethereum's privacy, scalability and security attributes.*[50)]

Common Core

The Common Core contains components that are used by both the ledger and ancillary data subset of components, as well as characteristics and attributes that apply to all components within the DIDO network. There are three classes of common components: tools and interfaces, distribution software, and secure messaging. Common components can be used exclusively by one kind of DIDO node as defined in node architecture but can also span across the components within a node. For example, the transaction API is available to both the ledger and the ancillary data node and potentially by some of the tools. However, the transaction API may or may not be used as part of the ancillary data stack.

By definition, a ledger operation node must rely on the transaction API to access the ledger; however, a smart contract node may also require access to the ledger. In that case, its access is made exclusively through the transaction API. It is up to the implementation of a particular DIDO whether to access its ancillary data through a transaction API or a transform API. Conversely, a transaction might require ancillary data (i.e., monetary exchange rate, stock quotes, interest rates, market cap, or Beta, etc.) in order to complete. The Common Core contains an API that defines and allows for this access, sometimes referred to as an oracle.

Note: Transaction API and Transform API are defined in more detail in Section 2.2.2.4 Messaging View.

- 2.2.2.3.1 Immutable Data Objects
- 2.2.2.3.2 Ancillary Data
- 2.2.2.3.3 Semantic Web
- 2.2.2.3.4 Software

49)

A. Hertig, “Does the original Bitcoin Wallet Still Matter?,” 16 September 2016. https://www.coindesk.com/original-bitcoin-wallet-still-matter/.

50)

R. Sharma, “What is the Byzantium Hard Fork in Ethereum?,” 7 March 2018. https://www.investopedia.com/news/what-byzantium-hard-fork-ethereum/

2.2.2.3.1 Immutable Data Objects

return to Node Architecture

An immutable data object is data whose value cannot be changed. Any required update is recorded as a new immutable data object with a link back to its parent(s). An immutable data object represents things either real or virtual. The representation can range from a simple scalar value to a complex data structure of values. The things or items data objects represent are real-world things such as commodities or information. Commodities include things like gold, silver, grains, and so on. Information includes things like certificates of birth, death, marriage, or stock. Data objects can also represent virtual or abstract things such as virtual coins, frequent flier miles, loyalty points, data rights, etc.

The immutable data object component is focused on the care and maintenance of distributed, global data objects usually stored in a ledger, the identifiers (i.e., accounts) associated with the ledger, the transactions used to make updates to the ledger entries, and the wallets that contain identifier (i.e., account) information for a user. When immutable data objects represent a currency such as cryptocurrencies, the identifiers represent accounts; however, when the immutable data objects represent data such as commodities, inventories, and public records, the identifiers may not represent accounts but unique identifiers of the thing.

2.2.2.3.1.1 Ledger

return to Immutable Data Objects

Typically, an immutable data object is stored as a ledger. The term *ledger* is a distributed object that has its root within accounting systems and is comprised of a series of entries that represent an account at different stages in its life. This includes its value, an operation to be performed on the value, and potentially another account that receives or sends an amount. Ledger entries are immutable, i.e., once an entry has been added to the ledger, it can never be modified. New versions of the entry exist, and transactions capture the changes required to move between the data entry values.

For example, if the original value for an entry was AAAA and a transaction wants to modify the value to BBBB, the original value remains in the ledger and a new entry is made with the value BBBB. The new entry points back to the previous value that contained AAAA. Thus, by following the chain of modifications backwards, the provenance and ultimately the pedigree of the value can be determined.

The ledger is not limited to account numbers but can have general identifiers which represent fungible data. For example, a library has books, but the books are referenced by an identifier, not an account. Patrons of the library may have identifiers which might look like accounts, but simply identify the patron.

The ledger exists at all nodes within the network. In the blockchain implementations of DIDO, all the values in the ledger for a particular entity will be identical when all the outstanding transactions contained within a block have been validated and verified and properly distributed throughout the network. There are alternatives to blocks and blockchains. For example, Hashgraphs which have been proven to also solve the Byzantine general problem asynchronously without having to use expensive Bitcoin PoW algorithms. [51)]

51)

G. Kinglsy, “Hashgraph vs. Blockchain Is the end of Bitcoin and Ethereum near?,” coincodex, January 2018. https://coincodex.com/article/1151/hashgraph-vs-blockchain-is-the-end-of-bitcoin-and-ethereum-near/.

2.2.2.3.1.2 Transactions

return to Immutable Data Objects

Transactions contain cryptographically signed data required to describe the creation, transfer, or destruction of fungible data representing an asset stored within the ledger. The transaction captures the change in state of the contents of the ledger. Currently, each implementation of Blockchain defines its own unique concept of a transaction which depends on the kind of fungible data stored in the ledger.

Transactions represent operations that can be performed on fungible data represented in the ledger. In a financial ledger, the money associated with an account is the fungible data and it is represented by the balance associated with an account associated with an entry in the ledger. Money can only be added to or deducted from an account. Note: The actual money is not stored in the ledger, only a balance representing money is stored in the ledger. A slightly higher level concept would be the transfer money from one account to another (i.e., deduct from account nnn1 and add to account nnn2).

2.2.2.3.1.3 Identities

return to Immutable Data Objects

Associated with items in the ledger are identifiers which associate fungible data represented in the ledger with accounts or associated with transactions. In classic financial Ledgers, these identifiers are generally account numbers that contain a balance (a tally) of the fungible data associated with the account and the entry number of the row within the ledger. For example, account `nnn1` has a balance of $50. Ledger Entry `ttt2` adds $25 to account `nnn1`.

The evolution of advanced symbologies has helped the securities industry grow, but the limitations and costs imposed by the closed systems have become more apparent as companies and institutions continue to integrate operations on a global scale. Proprietary symbology now stands as one of the most significant barriers to increased efficiency and innovation in an industry that sorely needs it. Moreover, the lack of common identifiers is a key roadblock to achieving the holy grail of Straight-through Processing (StP). [52)]

52)

Object Management Group (OMG), “Financial Industry Global Identifier® (FIGI™), v1.0,” December 2015. http://www.omg.org/spec/FIGI/1.0/.

2.2.2.3.1.4 Wallets

return to Immutable Data Objects

Wallets are not quite like physical wallets carried around in pockets or handbags. They are repositories where the cryptographic keys associated with the fungible data managed by the ledger are stored. These keys are required to unlock access to the fungible data in the ledger. For example, within the Bitcoin Blockchain, there is an identifier associated with each Bitcoin. To use the Bitcoin, a public and a private key are required. The private keys are stored in the wallet.

2.2.2.3.2 Ancillary Data

return to Node Architecture

Ancillary ddta is focused on data that is not directly part of the fungible data stored in the ledger but is used in the support, care, and maintenance of the fungible data. Ancillary data is global in nature (all nodes need access to it) and is like the fungible data in the ledger, distributed in nature (all nodes have copies of the data). Some DIDO implementations may provide access to ancillary data through the use of oracles; however, if the ancillary data is not also implemented as a DIDO but as frontend to centralized or even decentralized data, many of the benefits of the distributed data are lost. For example, if the exchange rate between currencies is not also distributed but offered from a server through an oracle, access to the exchange data becomes a vulnerability to the use of the cryptocurrency data when an exchange rate is required.

Access to ancillary data depends on the source of the data. Some implementations will store all the ancillary data on the centralized server, others will provide access to ancillary data stored on a series of decentralized servers, while others will provide access to the data as fully decentralized networks (i.e., other DIDOs). Ancillary data is accessed through an oracle.

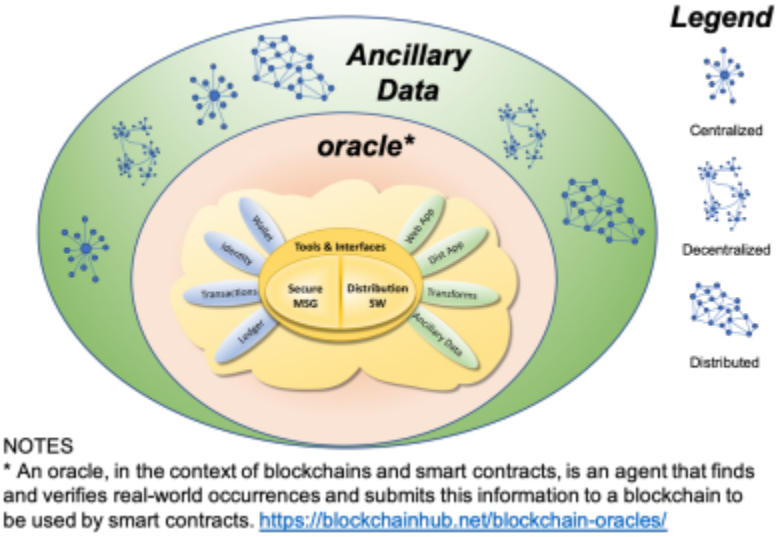


Figure 17: Relationship of Ledger, Ancillary Data and Oracles to other Network Topographies

The ancillary data, when implemented as a DIDO, is hosted in its own network of nodes which may or may not be the same set of nodes as the fiduciary data. It is comprised of the ancillary data itself, a mechanism for transforming the data (i.e., not necessarily a ledger transaction), and software that is distributed on all the nodes in its network, called web applications, which may or not run on a node within the network.

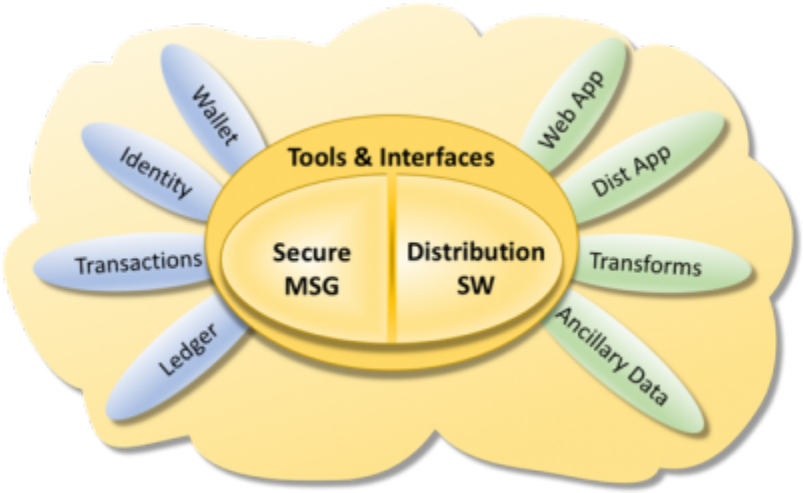


Figure 18: The Components in the Ancillary Data Stack of the DIDO Node

- 2.2.2.3.2.1 Journal
- 2.2.2.3.2.2 Transforms
- 2.2.2.3.2.3 Distributed Applications
- 2.2.2.3.2.4 Web Applications
- 2.2.2.3.2.5 Exchanges

2.2.2.3.2.1 Journal

return to Ancillary Data

The journal is data that is ancillary data to the main fungible data. It may or may not be implemented using a separate ledger and it may exist within another DIDO network, but its care and maintenance does not have to be part of the ledger's transactions. For example, the monetary exchange rate between two currencies is ancillary to a cryptocurrency ledger; however, the almost instantaneous updates to the exchange rate would not be appropriate in the cryptocurrency ledger itself.

2.2.2.3.2.2 Transforms

return to Ancillary Data

There are different ways ancillary data can be transformed. If the ancillary data is implemented as another “blockchain” DIDO, then the ancillary data is stored within another, parallel ledger and then, naturally, the transforms would be accomplished using that ledger stack’s transactions.

However, another mechanism available to transform the ancillary data is operational transformation (OT). These are similar to transactions but there can be multiple changes made to the data at the same time by multiple parties. Ancillary data using operational transforms is not necessarily immutable.

2.2.2.3.2.3 Distributed Applications

return to Ancillary Data

One of the major extensions to the original Satoshi Nakamoto [1] [27] papers on peer-to-peer electronic cash systems is the use of oracles to access external data and smart contracts to enforce business rules on ledger transactions. For example, an account holder wants to transfer Bitcoins from one account to another, but there is a “reserve” placed on the account to cover potential debts from option trading. This “reserve” information would not be part of the ledger, but would be ancillary data. The enforcement of the “reserve” is done by a distributed application called a smart contract.

> *A distributed application is software that is executed or run on multiple computers within a network. These applications interact in order to achieve a specific goal or task. Traditional applications relied on a single system to run them. Even in the client-server model, the application software had to run on either the client, or the server that the client was accessing. However, distributed applications run on both simultaneously.*
>
> *With distributed applications, if a node that is running a particular application goes down, another node can resume the task.* [53)]

A distributed application (DApp) is software that is executed or runs on multiple computers within a network simultaneously. The software is deterministic meaning that given the same inputs, they all produce the same outputs. One of the main benefits of DApps is they are extremely durable and hardened against a single point of failure. Therefore, to be distributed, deterministic and durable, any services that the DApp must interact with must also be distributed, deterministic, and durable. This makes DApps interaction with traditional client/server cloud services difficult. Naturally, cloud services applications such as Software as a Service (SaaS) and Data as a Service (DaaS) have some redundancy and reliability built into them; however, they cannot achieve the same level of robustness as a DApp. Therefore, SaaS and DaaS need to expose their functionality using a proxy DApp, which by its nature has latency. This latency could adversely affect the DApp’s deterministic and durability nature.

Another important aspect of a DApp is that it should be runtime environment agnostic: allowing as many DIDO nodes into the DIDO network as possible. This can be achieved using virtual machines such as a Java Virtual Machine (JVM) or interpretive engines such as those available with ECMAScript. There are some platform specific virtual machines available such as the Common Language Runtime (CLR) which are not standardized and do not run with the deterministic rigor on non-Windows platforms. Consequently, the list of languages allowable in DApps is limited to those that have a standardized runtime environment (i.e., VM or Engine) that runs on multiple platforms.

53)
Techopedia, “Techopedia Definitions,” 1 December 2017. https://www.techopedia.com/definition/23971/distributed-application.

2.2.2.3.2.4 Web Applications

return to Ancillary Data

Web applications are the main interface between the human end-users of a DIDO and the rest of the DIDO. The web applications do not have the same requirements of determinism as defined in the DIDO and can be subject to far less stringent timing requirements. The equivalent of a web application in the ledger stack is a wallet.

> *A web application or "web app" is a software program that runs on a web server. Unlike traditional desktop applications, which are launched by your operating system, web apps must be accessed through a web browser.* [54)]

54)

TechTerms, "TechTerms Definitions," 1 December 2017. https://techterms.com/definition/web_application.

2.2.2.3.2.5 Exchanges

return to Ancillary Data

Investopedia defines an Exchange as:

An exchange is a marketplace in which securities, commodities, derivatives and other financial instruments are traded. The core function of an exchange is to ensure fair and orderly trading and the efficient dissemination of price information for any securities trading on that exchange. Exchanges give companies, governments and other groups a platform from which to sell securities to the investing public. [55)]

Although this definition is meant to be specific to existing physical and electronic exchanges covering "securities, commodities, derivatives and other financial instruments" offered on the New York Stock Exchange (NYSE), Nasdaq, London Stock Exchange (LSE), and Tokyo Stock Exchange (TSE), it is also applicable when immutable data objects themselves become financial instruments and when they are applied to public records, certificates, or supply chains.

An exchange on the surface may seem to be a trivial concept and if this were so, the development of laws, rules, and regulations governing them would also be trivial. However, a quick look at the rules governing the NYSE alone is daunting [56)] . These rules are not to torment or torture traders, or to limit trades; rather they are to "ensure fair and orderly trading".

As the DIDO exchange concepts grow and mature, the need for, and the development of, rules and regulations to "ensure fair and orderly trading" must also adopt the rules and regulations expected from any exchange and evolve to handle the unique benefits, functions, capabilities of DIDOs.

Some of the types of trades that an exchange could handle are:

- **Market Order (MKT)** – A market order is an order to buy or sell a stock at the best available price. Generally, this type of order will be executed immediately. However, the price at which a market order will be executed is not guaranteed. It is important for investors to remember that the last-traded price is not necessarily the price at which a market order will be executed. In fast-moving markets, the price at which a market order will execute often deviates from the last-traded price or "real time" quote. [57)]
- **Limit Orders (LMT)** – A limit order is an order to buy or sell a stock at a specific price or better. A buy limit order can only be executed at the limit price or lower, and a sell limit order can only be executed at the limit price or higher. A limit order is not guaranteed to execute. A limit order can only be filled if the stock's market price reaches the limit price. While limit orders do not guarantee execution, they help ensure that an investor does not pay more than a pre-determined price for a stock. [58)]
- **Stop Order (STP)** – A stop order, also referred to as a stop-loss order, is an order to buy or sell a stock once the price of the stock reaches a specified price, known as the stop price. When the stop price is reached, a stop order becomes a market order. A buy stop order is entered at a stop price above the current market price. Investors generally use a buy stop order to limit a loss or to protect

a profit on a stock that they have sold short. A sell stop order is entered at a stop price below the current market price. Investors generally use a sell stop order to limit a loss or to protect a profit on a stock that they own. [59)]

- **Stop Limit Order (STPLMT)** – A stop-limit order is an order to buy or sell a stock that combines the features of a stop order and a limit order. once the stop price is reached, a stop-limit order becomes a limit order that will be executed at a specified price (or better). The benefit of a stop-limit order is that the investor can control the price at which the order can be executed. [60)]
- Some of the types of trades that need to be considered during an Exchange are **Fill-Or-Kill (FOK)** – An FOK order is an order to buy or sell a stock that must be executed immediately in its entirety; otherwise, the entire order will be cancelled (i.e., no partial execution of the order is allowed). [61)]
- **All-Or-None (AON)** – An Aon order is an order to buy or sell a stock that must be executed in its entirety, or not executed at all. However, unlike the FoK orders, Aon orders that cannot be executed immediately remain active until they are executed or cancelled. [62)]
- **Market If Touched (MIT)** – A market-if-touched, or MIT, order is a conditional order that becomes a market order when a security reaches a specified price. When using a buy market-if-touched order, a broker will wait until the security falls to a certain level before purchasing the asset. A sell market-if-touched order will activate when the price of a security rises to the specified level. [63)]

This document provides a list of standards associated with each of the components within the DIDO Reference Architecture. In this section the descriptive text for each standard or specification is taken directly from the original standard or specification and is properly attributed with a reference to that standard or specification. This has been done in order to aid readers of the DIDO RA to be able to determine applicability and usefulness of the standard or specification to the particular instance of a DIDO they are working on.

Re-writing the descriptive text can result in a misunderstanding of the original intent of those standards and specifications. If you have ever been involved in the writing of a standard or a specification, you'll be familiar with the long discussions and debates on the selection of each word and the punctuation used. Therefore, the text in the standards listed below is duplicated as much as possible "intact".

55)
Investopedia, "Exchange,". https://www.investopedia.com/terms/e/exchange.asp#ixzz5OGeT4QUj.
56)
New York Stock Exchange, "NYSE Tools," http://wallstreet.cch.com/NYSETools/PlatformViewer.asp?selectednode=chp%5F1%5F2%5F1%5F35&manual=%2Fnyse%2Frules%2Fnyse%2Drules%2F.
57)
U.S. Securities and Exchange Commission, "Market Order," https://www.sec.gov/fast-answers/answersmktordhtm.html.
58)
U.S. Securities and Exchange Commission, "Limit Orders," SEC, https://www.sec.gov/fast-answers/answerslimithtm.html.
59)
U.S. Securities and Exchange Commission, "Stop Order," SEC, https://www.sec.gov/fast-answers/answersstopordhtm.html.
60)
U.S. Securities and Exchange Commission, SEC,

https://www.sec.gov/fast-answers/answersstoplimhtm.html.
61)

U.S. Security and Exchange Commission, SEC,
https://www.investor.gov/additional-resources/general-resources/glossary/fill-or-kill-order.
62)

U.S. Security and Exchange Commission, SEC
https://www.investor.gov/additional-resources/general-resources/glossary/all-or-none-order.
63)

Investopedia, Market If Touched - MIT,
https://www.investopedia.com/terms/m/marketiftouched.asp#ixzz5OHOMW2ba]].

2.2.2.3.3 Semantic Web

return to Node Architecture

A major promise of DIDO is the ability to use machines to automatically process data and transactions rather than to rely on humans “in the middle.” A key to automating many of these processes is realized through the adoption of the semantic web. In essence, the semantic web is the institutional memory and experience captured in machine readable form and available at each node. This eliminates the centralized human-centric requirements of the past and supports a distributed, automated solution. The semantic web uses formal semantics that unambiguously capture the vocabulary and ontology of the domain.

2.2.2.3.4 Software

return to Node Architecture

By their very nature, DIDOs are run on distributed, decentralized computers that require software to operate. The DIDO software itself can be classified as ancillary data and therefore can be distributed just like data to each DIDO node. The DIDO software could include smart contracts, distributed applications (DApps), scripts, containers, and DIDO command line interface (CLI) commands.

2.2.2.4 Messaging View

return to Node Network View

There are three classes of messages that are used within the DIDO node network.

Transactions

Transactions are messages containing instructions on how to change the ledger from one known state to the next state. Transactions are sent to all the nodes within the Node Network providing instructions on how to modify the ledger data to the next state. When all the transactions have been applied to all ledgers on all of the individual nodes within the node network, the ledgers will have the same state and can be treated as a single datastore. These transactions are executed using a transaction Application Programming Interface (API), an API.

Transforms

Like transactions, transforms are instructions that change the state of a “ledger” from one state to another. Transforms are not as order dependent as transactions and are NOT sent to all the nodes. Transforms may or may not be sent to DIDO Nodes. The Transform Network represents Transform Nodes participating in the transformation of some content. Once the content transformation is complete, the changes made from the initial state of the content to the new state of the content are formulated into a Transaction and sent to all the Nodes in the DIDO Network. A Transform API is used to perform this action.

Streams

Streams are a way of subsetting or filtering DIDO Transactions using a publish/subscribe paradigm. This allows any particular Node within the Node Network not to have to listen-to or process-all the transactions presented to it. This is similar to Transforms; however, the participants in the transforms do not have to be part of the Node Network.

2.3 Taxonomic Views

return to Architectural Views

A taxonomy is a way of organizing things into useful and convenient classes. Each class can then be treated as an abstraction of the individual elements. For example, **Dog** is a class of **Animal**. Once an individual entity is classified as a **Dog**, generalizations can be made about what to expect from the individual entity. There can be many taxonomies that classify entities. For example, one taxonomic classification places an individual in the animal kingdom; however, another taxonomy classifies an individual according to their state of employment (working, retired, etc.). Each is valid; the only difference is perspective.

The DIDO RA employs the following four taxonomies:

2.3.1 Network Topology Taxonomy

return to Taxonomic Views

Probably one of the most significant differences between DIDO architectures and other architectures is the simple, but powerful, topology of the network of nodes it uses to connect peers. DIDOs rely primarily on a distributed rather than a decentralized or centralized topology. Each network topology defines a community of nodes that act as peers, which collectively provide a solution to a problem that is then distributed. This community of nodes also employs redundant data storage, redundant computing, or both.

Most DIDO implementations rely heavily on data and computational power that exists externally and, therefore, is beyond the primary focus of the DIDO. For example, an account holder in a cryptocurrency DIDO application may require information such as currency exchange rates, the holder's nation of origin, tax IDs, or certificates of trust. In a greenfield development, all this external data would be held within the DIDO. The reality is that it is not possible to build everything from scratch. Therefore, this data might be held in other network topologies (see section 2.3.4.2 Ancillary Data).

These three kinds of network topologies are represented graphically in the following figure.

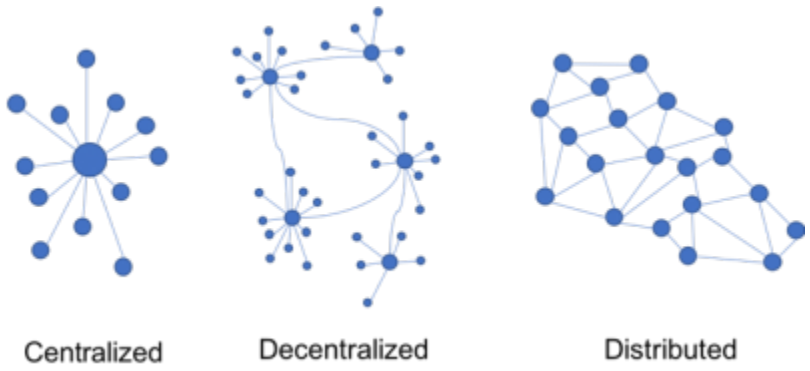


Figure 19: The Difference between Centralized, Decentralized, and Distributed Network Topologies

- 2.3.1.1 Centralized Network Topology
- 2.3.1.2 Decentralized Network Topology
- 2.3.1.3 Distributed Network Topology

2.3.1.1 Centralized Network Topology

return to Network Topology Taxonomy

The centralized network topology (usually associated with mainframes) has a very large centralized server node that all other nodes connect to. It provides most of the services such as data storage, processing, backups, and computational power for the other nodes. It is possible to implement a “ledger” using the centralized model. In many ways, the centralized model that holds a single ledger is easier to implement and maintain since there is only one version of the data in one place and consequently, by definition, the data is the canonical data (i.e., authoritative or standard data).

Figure 20: Centralized Network Topology

Trying to remove redundant, replicated, or duplicate data is a fundamental principle of the centralized model. The multiple “copy of data” problem was first formally addressed by E. F. Codd in his development of A Relational Model of Data for Large Shared Data Banks[64] and has become the cornerstone of Relational Database Management Systems (RDBMS), used extensively today in products such as Oracle Corporation’s Oracle, IBM’s DB2, or Computer Associates INGRES [65] . The following chart shows that the centralized database market is not shrinking, and by 2017 had grown to a net worth of $50 billion with no signs of change in this growth trend. [66] This highlights the magnitude of the effort to convert the world to DIDO technology. Though it may eventually happen, it’s going to take a long time.

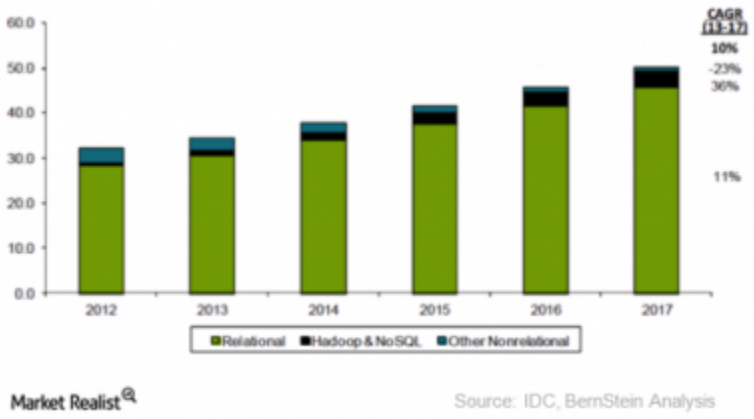


Figure 21: Centralized Global Database Market ($ Billions)

64) E. F. Codd, “A relational model of data for large shared data banks,” Communications of the ACM, vol. 13, n. 6, pp. 377-387, June 1970

65) G2 Crowd, “Best Relational Databases Software in 2018,” 2018. [Online]. Available: https://www.g2crowd.com/categories/relational-databases [Accessed 12 June 2018].

66)

A. Shields, “Is Oracle's Position Secure in the Database Space?,” 18 January 2016. [Online]. Available: http://marketrealist.com/2016/01/oracles-position-secure-database-space/. [Accessed 22 July 2017].

2.3.1.2 Decentralized Network Topology

return to Network Topology Taxonomy

In a decentralized network topology, there are several “central” nodes, each providing redundancy and failover capabilities for the other. Often, each of the centralized nodes are geographically distributed (i.e., North American, Europe, Southeast Asia, etc.).

Figure 22: Decentralized Network Topology

Decentralized systems overcome some of the problems associated with a centralized system: they do not have a single point of failure, resulting in fault tolerance.[67),68)] Thus, each node can be maintained independently, which ultimately results in a more stable system. Moreover, the load on the system can be reduced by increasing the number of centralized nodes thereby distributing the work load.

Nevertheless, the scalability of the system is moderate, since the cost of expansion per node is generally steep. Granted, much of this cost has come down with the availability of Platform as a Service (PaaS) offerings from Amazon, Microsoft, and others. Since the nodes are arranged in clusters around one of the centralized servers, there is only partial fault tolerance with the result that occasionally parts of the system are rendered unavailable. The dependence on the underlying network topology means that, depending on which network links are broken, there is a chance that the data within one of the servers is obsolete.

The decentralized model is used extensively in cloud computing, specifically Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform-as-a-Service (PaaS). Although there are implementations of cloud computing which do not use the decentralized model (e.g. blockchains), the following chart from Gartner summarizes the projected growth of the “as a Service” offerings and indicates that it is on the rise. [69)] It also highlights the magnitude of trying to “convert” the world to one that is completely distributed. Even were it to occur, it’s going to take a long time.

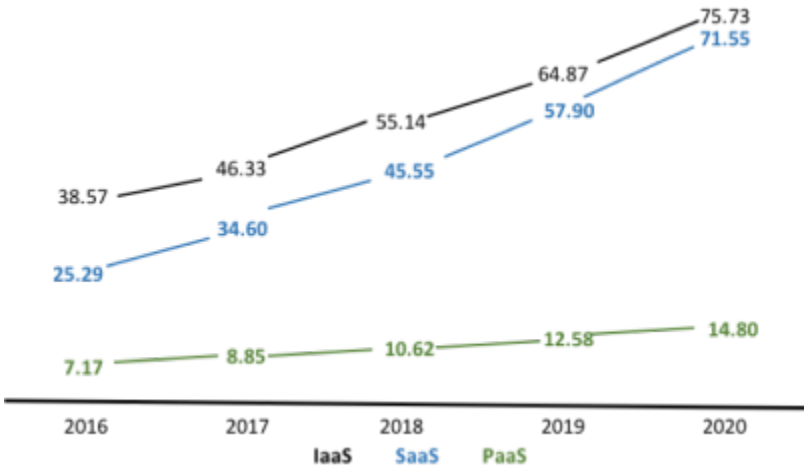


Figure 23: Decentralized Cloud Market Revenue ($ Billions)
67)

Nonetheless, the October 2016 Distributed Denial of Service (DDoS) attack on Domain Name System (DNS) servers on the US East coast and in Europe highlighted a weakness in the decentralized model.
68)

A. Shields, “Is Oracle's Position Secure in the Database Space?,” 18 January 2016. [Online]. Available: http://marketrealist.com/2016/01/oracles-position-secure-database-space/
69)

C. Coles, “Overview of Cloud Market in 2017 and Beyond,” 2016. [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/. [Accessed 24 July 2017].

2.3.1.3 Distributed Network Topology

return to Network Topology Taxonomy

In the distributed network topology, all nodes are equal peers within the system, with each acting as a redundant copy of other nodes. This provides for extremely high fault tolerance, tied to the number of nodes. The more nodes, the greater the tolerance to faults. Since each node is an equal peer, there is no single point of failure and there is no one master copy of the data. As a result, the distributed system becomes almost infinitely scalable.

Figure 24: Distributed Network Topology

However, all nodes are not really considered “equal” in the truest sense. Some nodes are used simply to create a simple transaction, others record all the information within the blockchain, and others go on to validate transactions by mining.

Figure 25: Market Cap of Cryptocurrencies ($ Billions)

2.3.1.4 Relevant Networking Standards

return to Network Topology Taxonomy

The following standards are applicable to all network views.

Technical Standards

- RFC0793 - Transmission Control Protocol
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC1112 - Host Extensions for IP Multicasting
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC0768 - User Datagram Protocol (UDP)

de facto Standards

- None identified

2.3.2 Network Access Control Taxonomy

return to Taxonomic Views

Network access control taxonomy classifies the types of access individuals (i.e., nodes) have from outside and from within the node network. The two main classes of access control are permissionless and permissioned.[70)]

Within each of these two classifications it is possible to have public and private access. Public and private access define who is able to write data onto a network or ledger. In contrast, open (i.e., permissionless) and closed (i.e., permissioned) determine who is able to read the data. Networks are classified as[71)]:

- public and open
- public and closed
- private and open
- private and closed

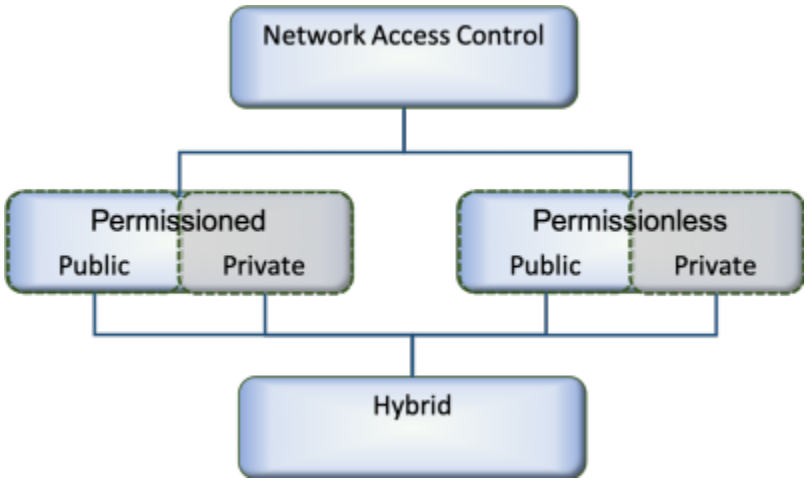


Figure 26: The Node Network Access Taxonomy

Another category of networks is a hybrid network, which makes it possible to restrict the visibility of information on the network using a combination of public, private, permissionless and permissioned networks. Therefore, hybrid networks are appealing to regulated markets because they offer the benefits of public blockchain and private blockchain together.[72)]

Permissionless Networks	Permissioned Networks
Public Networks	Private Networks
Hybrid Networks	

70) , 71)
“Public Vs Private Blockchain In A Nutshell”, Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

72)
“Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.”, Mina Down, 14 November 2018

https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

2.3.2.1 Permissionless Networks

return to Network Access Control

Permissionless Networks require no permission to use them. In other words, there is no barrier to entry. Anyone can run a node, run mining software/hardware, access a wallet, and write data onto and transact within the blockchain (as long as they follow the rules of the blockchain). There is no way to censor anyone, ever, on a permissionless Bitcoin blockchain.[73)]

Benefits of Permissionless Networks

Decentralized

Permissionless networks are decentralized and distributed. In other words, no one entity can close or terminate the network, modify the content, or censor parts of it. The larger the distributed and decentralized networks and or history are, the harder it is to tamper with.[74)]

Transparency

Users or nodes have complete access to the ledger, transactions, and blocks in the blockchains, which allows for complete auditing of permissionless networks[75)].

Anonymity

In permissionless networks, users or nodes of the network are anonymized. Technically, permissionless networks like Bitcoin are pseudonymous, and not truly anonymous.[76)]

Governance

As a general rule, permissionless networks rely on open source software, which is ruled by open source communities (see Talk Openly Develop Openly (TODO)). The governance of the network is by consensus. Consensus is different for many of the permissionless networksm(i.e., Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), etc). [77)]

Tokens

Permissionless blockchains employ fat protocols that compensate network contributors with

Tokens. As the value and utility of the network increases, the value of the underlying tokens increases as well. This is the premise of cryptoeconomics and Initial Coin Offering (ICO) based fundraising. There are two predominant types of tokens today: monetary value tokens and utility tokens. Monetary value tokens are used in myriad ways as instruments for exchanging value. Utility tokens are akin to loyalty points: they have intrinsic value but no monetary value outside of that ecosystem.[78)]

Scalability and Performance

For all the value blockchains bring to modern business processes, their Achilles heel often involves scalability and performance. Both Bitcoin and Ethereum blockchains suffer from poor scores in this area. For example, a recent blockchain game called Crypto kittles clogged the Ethereum network. Having said that, these are just early teething troubles, and startups are experimenting with various strategies to address this issue. Hopefully it is only a matter of time before this issue becomes a non-entity.[79)]

[73)]

“Permissioned vs. Permissionless blockchains: Who will win and will it matter?”, Dustin D, 22 March 2018, Permissionless

[74)], [75)], [76)], [77)], [78)], [79)]

“Nuances Between Permissionless and Permissioned Blockchains”, Anant Kadiyala, 18 February 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483

2.3.2.2 Permissioned Networks

return to Network Access Control

Permissioned Networks (Glossary Definition)

Permissioned blockchains combine the properties of both the public network and private network. Each permissioned network is unique and represents a careful balance of public and private networks to meet specific business use cases.

The available options include allowing anyone to join the permissioned network after suitable verification of their identity, and the allocation of select and designated permissions to perform only certain activities on the network. [80)]

Benefits of Permissioned Networks

Decentralization

The degree of decentralization for permissioned networks is a business decision. The extent and quality of decentralization depends upon the number of peers (i.e., nodes), the expected number of bad nodes in the network, and the type of consensus mechanism determined by the stakeholder. Permissioned blockchains usually employ an algorithm such as Byzantine Fault Tolerance, which differs from the proof of work (PoW) algorithm[81)].

Transparency

Transparency is not a driving force in permissioned networks and is often a major factor in the business decision to choose permissioned over permissionless networks. Most permissioned blockchains do not use cryptoeconomic coins incentive or tokens. The primary incentive of permissioned blockchain participants is to minimize the transparency, cost, time, and ease of sharing information[82)].

Privacy

Permissioned blockchains offer fine-grained visibility into transaction details, as well as, metadata about those transactions which, in many ways, compromises the privacy of the Network participants[83)].

Governance

There are fundamental differences between permissionless and permissioned network governance. Permissioned governance is decided and agreed upon by members of the business network. Economic incentives, code quality, code changes, and power allocation among peers are based on the business dynamics and the common purpose and goals of the permissioned members. This allows for agile and responsive networks desired by businesses[84)].

Tokens

Permissioned blockchains generally do not employ a cryptoeconomic coins incentive or tokens[85)].

Scalability and Performance

Permissioned blockchains use consensus mechanisms, which are computationally inexpensive (when compared to proof of work (PoW)). Therefore, they enjoy substantially better scalability and performance than their permissionless network cousins[86)].

80)
"Public, Private, Permissioned Blockchains Compared", Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/
81) 82) 83) 84) 85) 86)
"Nuances Between Permissionless and Permissioned Blockchains", Anant Kadiyala, 18 February 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483

2.3.2.3 Public Networks

return to Network Access Control

Public networks are distributed and open, allowing participation in core activities of the network from any node sponsored by anyone. Participation includes joining, leaving, reading, writing, and auditing the activities on the network. There are no authorities or discrimination of nodes.

Benefits of Public Networks

Open Read and Write[87)]

Anyone can participate by submitting transactions to the blockchain, such as Ethereum or Bitcoin; transactions can be viewed on the blockchain explorer.

Ledger Is Distributed[88)]

The database is not centralized like in a client-server approach, and all nodes in the blockchain participate in the transaction validation.

Immutable[89)]

When something is written to the blockchain, it can not be changed, in other words it is immutable.

Secure Due to Mining (protection from the Fifty-One Percent (51% Attack)[90)])

For example, with Bitcoin, obtaining a majority of network power could potentially enable massive double spending, and the ability to prevent transaction confirmations, in addition to other potentially malicious acts.

87)
"Public Vs Private Blockchain In A Nutshell", Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f
88), 89), 90)
"Public Vs Private Blockchain In A Nutshell", Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

2.3.2.4 Private Networks

return to Network Access Control

A **private network** limits access to the network to individuals or nodes granted and verified to have permission to participate in joining, leaving, reading, writing, and auditing activities on the network. A participant joins a private network only through an authentic and verified invitation; validation is performed either by the network operator(s) or using a clearly defined set of protocols implemented by the network.[91)]

Benefits of Private Networks

Enterprise Permissioned[92)]:

The enterprise controls the resources and access to the blockchain, hence private and/or permissioned.

Faster Transactions[93)]:

When you distribute the nodes locally, but also have far fewer nodes that participate in the ledger, performance is faster.

Better Scalability[94)]:

Being able to add nodes and services on demand can provide a great advantage to the enterprise.

Compliance Support[95)]:

As an enterprise, you would likely have compliance requirements to adhere to; having control of your infrastructure enhances ability to satisfy this requirement more seamlessly.

Consensus More Efficient (fewer nodes)[96)]:

Enterprise or private blockchains have fewer nodes and usually a different consensus algorithm, such as BFT vs PoW.

91)

“Public, Private, Permissioned Blockchain Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

92) , 93) , 94) , 95) , 96)
“Public Vs Private Blockchain In A Nutshell”, Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

2.3.2.5 Hybrid Networks

return to Network Access Control

A **Hybrid Blockchain** is unique in that it is decentralized while also making it possible to restrict the visibility of information on the network with a combination of public, private, permissionless and permissioned Networks. Thus, a hybrid blockchain is appealing for regulated markets as it offers the benefits of public blockchain and private blockchain together.[97)]

The hybrid blockchain's decentralized, secure, transparent, and immutable nature provides benefits similar to those offered by permissionless and public networks: it allows for restrictions on rights to view, modify, and append/approve transactions to approved participants. In simple words, if a network member does not want their transaction data to be visible or accessible without their permission, they can earmark particular rights to view, modify, or get into consensus with different members.[98)]

Benefits of Hybrid Networks

Private Transactions

Transaction are private but verifiable using the ledger's immutable data objects (i.e., leverage its public state). In its public state, each transaction gets approved by a massive network and is essentially secure and trustworthy. Hence, there is no need for a central governing body or an exhaustive chain of intermediaries to supervise things. So, any change done to a transaction will undergo a “kindred” approval process, making it next to impossible for a single actor to meddle with the transaction or entries[99)].

Equality

Everyone in the network has equal rights to view, modify, and append their consent to a transaction. In addition, the identity of transacting parties is never disclosed to all the visible network participants. [100)].

Non-Repudiation

Anonymity is simply not acceptable to financial institutions and regulated industries with their strict Know Your Customer (KYC) standards. [101)].

Confidentiality

Unrestricted visibility of the public state of the network exposes all the data to a colossal network breach, which is counter to data confidentiality obligations, as well as their business concerns.[102)].

97)

“Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.”, Mina Down, 14 November 2018
https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

98) , 100) , 101) , 102)

“If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain”, Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794

99)

“If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain”, Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794. This article uses the term “agnate approval” rather than “kindred approval”; however, agnate limits a kindred relationship to males only. Thus, we prefer the term “kindred” over “agnate”

2.3.3 Node Taxonomy

return to Taxonomic Views

A node[103] is any participant in a DIDO Node Network. However, there are different types of nodes within the node network. The types are classified according to the kind of activities (i.e. roles) they need or want to perform within the node network.

All nodes must provide their own hardware to participate. The hardware requirements can range from fairly simple handheld devices to larger servers capable of storing vast amounts of data and processing many transactions. However, servers are not essential for the operation of the node network since all participants operate as peers within a peer-to-peer (P2P) network of equals (i.e., it is not a client-server model). The correct value of data (i.e. “Truth”) is achieved through consensus and results in each node having a local copy of the “true” values. Thus, there is no need to go to a central server to get the true, accurate, or current values of the data.

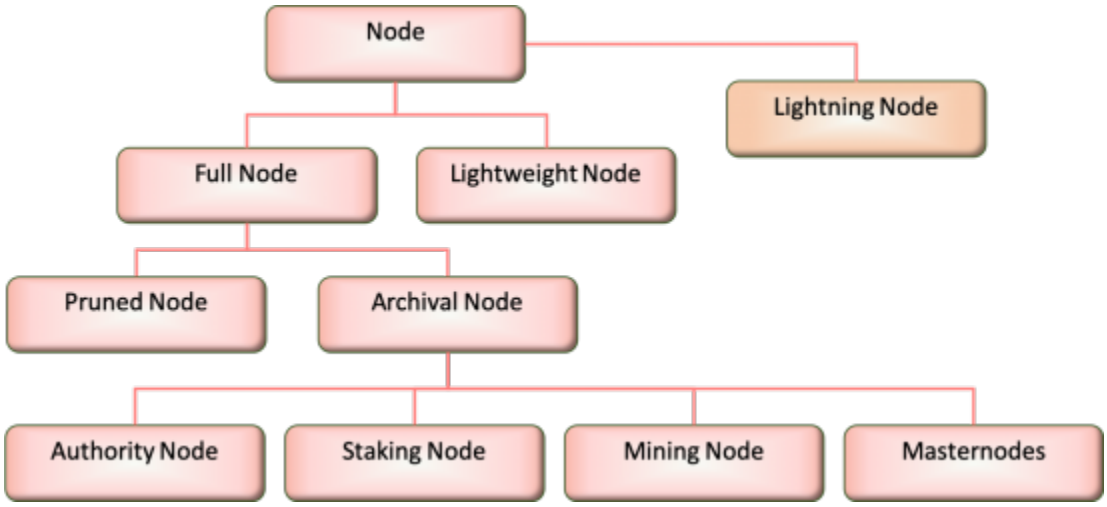


Figure 27: DIDO Node Taxonomy: Node Types

The classifications shown in Figure 1 are generalizations; each domain may define more or fewer types of nodes with each assigned different roles than those described here. For example, Bitcoin really defines only two kinds of nodes: full nodes and lightweight nodes. Full nodes have the entire copy of the ledger and can create transactions on their own. Lightweight nodes must work with a full node in order to synchronize the current “true” value of the data. Bitcoin also has a mining node, which is a full node used to validate that a block of transactions is “true.” Sometimes DIDO nodes can be referred to as clients[104] and are classified by their level of engagement with the DIDO; however, any domain can define or modify the definitions for node types.

Full Node			Lightweight	Lightening
Pruned	Archival			Permanode
	Authority	Staking	Mining	Master

103)

“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

104)

Clients are nodes able to parse and verify blockchain ledger, transactions, and smart contracts. Clients have access to APIs with which to create transactions and mine blocks.
https://ethereum.stackexchange.com/questions/269/what-exactly-is-an-ethereum-client-and-what-clients-are-there

2.3.3.1 Full Node

return to Node Taxonomy

Full nodes keep a full copy of the blockchain transactions[105]. There can be any number of full nodes within the node network, all acting as redundant data sources. Some of the activities of a full node are maintaining consensus between other nodes, verification of transactions, and storing the ledger.

In many ways the full nodes' functionality is analogous to those of servers in decentralized networks. However, there is no centralized “truth” or final judge. Instead, the “truth” is determined by consensus among the full nodes. However, this consensus based methodology is not without its pitfalls. When more than 51% of the full nodes cannot reach consensus (i.e., agree with a transaction or a proposition), the proposed change is skipped. This can lead to a Hard Fork in the ledger and the opposing groups diverge, creating two or more chains [106]. Sometimes the 51% problem can be part of an orchestrated effort, referred to as a 51% attack [107]. The more nodes in the node network, the harder it is to successfully launch a 51% attack.

A well-known example of this kind of ledger divergence, leading to a hard fork, was the Bitcoin Cash Fork.[108]

Full node contains:

- 2.3.3.1.1 Pruned Node
- 2.3.3.1.2 Archival Node

105)
Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7
106)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/
107)
“51% Attack”, Jake Frankenfield, 6 May, 2019, https://www.investopedia.com/terms/1/51-attack.asp
108)
“Bitcoin Cash’s Scheduled Hard Fork Tripped Up By Software Bug”, Christine Kim, 15 May 2019, https://www.coindesk.com/bitcoin-cash-scheduled-hard-fork-tripped-up-by-software-bug

2.3.3.1.1 Pruned Node

return to Full Node

A **pruned node** is a full node for all intents and purposes with one major difference: in order to save space on the node, the pruned node begins downloading blocks from the beginning of the ledger and when a preset threshold on space is exceeded, the oldest transactions are deleted, retaining only the headers and placement within the blockchain.

For example, a size limit threshold of 550MB allows storage of the latest blocks that fit within the 550MB threshold. However, to retain the integrity of the ledger, all the transactions must be processed and checked for validity in the order of their creation. At the end of the load, the state of the data is correct and only the history is of the transactions is lost.[109)]

Note: Pruned nodes are considered full nodes and thus can also verify transactions and be involved in consensus.

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

109)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

2.3.3.1.2 Archival Node

return to Full Node

In more traditional DIDOs, an **archival full node** is primarily responsible for storing the entire ledger and for providing consensus to the entire node network by validating blocks and potentially receiving a reward [110)].

As the size of the ledgers grow, there will be fewer archival nodes and more pruned nodes. IOTA refers to these archival nodes as “permanodes” and believes that business services will evolve around either charging for the storage or charging to retrieve the data that is stored or perhaps both. In some cases, such as in the Industrial Internet of Things (IIoT), the nodes that publish data might be rewarded when or if the the data is retrieved.

> **Note:** The difference between pruned and archival nodes is that pruned nodes only keep the latest validated data and require far less storage on the local node.

Archival nodes are divided into subtypes, three that can add blocks to a blockchain and one that cannot:

- Can add blocks: authority node, staking node, and mining node
- Cannot add blocks: masternode

110)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/. Article covers the various kinds of “rewards” a node can receive. Descriptions of the Archival Node subtypes also covers rewards.

2.3.3.1.2.1 Authority Node

return to Archival Node

Traditionally, the participation in DIDO node networks to perform tasks is permissionless, requiring no outside authority from anyone or anything. Having permissionless access to a node network is one of the original guiding principles that is integral to the decentralized nature of DIDOs[111].

> **Note:** Unfortunately, there are drawbacks to this approach. Solutions involving a level of centralization for granting permission can provide benefits like increased speed because there is no need for the costly, time consuming consensus algorithms such as Delegated Proof of Stake (DPoS), Delegated Byzantine Fault Tolerant (dBFT), Proof of Authority (PoA) and others. However, centralized permissioned systems are vulnerable to malicious attacks such as denial of service, thereby undermining many of the “democratizing” aspects of the DIDO.

Networks making use of PoA networks define a fixed number of **authority nodes**. The number of nodes and associated PoA designation is voted on by the PoA community or defined by the development team. Authority nodes are similar to full nodes and can create and validate blocks. All other nodes in the node network run as lightweight nodes, depending on broadcasted data to participate in the blockchain.[112]

Iota refers to authority nodes as **permanodes** (definition) because they keep all transaction data even after snapshots are made (concept introduced by Iota [113])

111)
S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

112)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

113)
“Full node vs permanode”, Helmar, 5 January 2018, https://iota.stackexchange.com/questions/782/full-node-vs-permanode/783

2.3.3.1.2.2 Staking Node

return to Archival Node

A **staking node** is usually a lightweight node (wallet). It allows for the aggregation of individual stakeholders stakes in Proof of Stake (PoS) a node network. The collection of tokens (i.e., coins) has a higher weight [114)] and improves the chances of receiving more rewards for validating a block of transactions. [115)]

Characteristics of Staking Nodes[116)]

- Ease of setup (basically a lightweight node (wallet))
- No initial startup costs
- No payout delay
- No penalty for being offline
- No minimum balance
- No guarantee of returns
- Usually smaller returns than a masternode

114)
See Weight of Network
115)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/
116)
“Masternode vs Staking”, Solaris Support Center, 26 June 2019, https://solaris.helpsite.com/articles/24861-masternode-vs-staking

2.3.3.1.2.3 Mining Node

return to Archival Node

Mining nodes [117) 118)] are full nodes or lightweight nodes within the node network that attempt to prove they have worked and completed the required challenge before anyone else to create a block of transactions (see proof of work (PoW)). To complete the task, mining nodes perform either as full archival nodes, or receive data from other full nodes on the node network to obtain the current state of the blockchain and parameters required to describe the next block.

Mining nodes employ hardware components (i.e., CPUs, GPUs or ASICs) to solve a cryptographic problem. The first mining node to complete the task broadcasts the results to the node network for verification by full nodes. Once consensus is achieved on the determination that the solution to the problem is correct and the solution is the first to be posted, the mining node is granted the right to add a block to the existing blockchain.

As a reward for performing the work, mining nodes are given a preset number of tokens (i.e., coins), as well as the transaction fees associated with the block of transactions. The preset reward is often referred to as a *coinbase* or a *coinbase transaction*. The reward (i.e., coinbase) is usually the first transaction in the block and free. [119),120)]

117)
Not all DIDOs use “mining” as originally defined in Bitcoin as Proof of Work (PoW). This means other DIDO implementations (i.e., IOTA) may not require a full blockchain Ledger to verify a transactions validity.
118)
S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.
119)
“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/
120)
Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7

2.3.3.1.2.4 Masternode

return to Archival Node

A **masternode**[121] is combination of a staking node using a Proof of Stake (PoS) node network, which relies on the weight of the stake[122], and a server.

The requirements for a successful staking node are a server, a stable internet connection, a minimum number of coins used for staking, and time to mine the server. Unlike the staking node, the masternode does not wait for randomly assigned blocks of transactions to validate but is instead constantly engaged: obtaining rewards and constantly paying out a certain number of tokens (i.e., coins) (thus the minimum stake).[123]

Characteristics of Masternodes[124]

- Higher rewards than a simple staking node
- Ability to participate in votes on proposals
- Hosting does not have to be local
- Higher cost and resource requirements than staking node
- More complexity
- Requires an initial stake

121)
"Blockchain Nodes: An In-Depth Guide", https://nodes.com/
122)
See Weight of Network
123)
"What's the difference between staking and masternode?", darkangel11, 03 February 2018, https://bitcointalk.org/index.php?topic=2874856.0
124)
"Masternode vs Staking", Solaris Support Center, 26 June 2019, https://solaris.helpsite.com/articles/24861-masternode-vs-staking

2.3.3.2 Lightweight Node (Wallet)

return to Node Taxonomy

Lightweight nodes keep a shallow copy of blockchain transactions[125)] and are used in day-to-day crypto operations. Within the cryptocurrency world, lightweight nodes are also referred to as simple payment verification (SPV) nodes or wallet nodes. Lightweight nodes communicate with the ledger through full nodes.[126)]

Standards

Technical Standards

- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)

- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Solidity Language Specification
- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- None at this time

125)

Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7

126)

“Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

2.3.3.3 Lightning Node

return to Node Taxonomy

A **lightning node** is a participant in a lightning network[127] using multi-signature (multisig)[128] on a payment channel[129]. Lightning nodes create a small community (2 to n) of blockchain nodes. The community can trade without having to use the expensive ledger transaction and has been proposed as a way to provide scalability to Bitcoin.

127)
"Blockchain Nodes: An In-Depth Guide", https://nodes.com/
128)
"How to Use Multisig to Keep Your Coins Ultra-Safe", Kai Sedgewick, 2 March 2019, https://news.bitcoin.com/how-to-use-multisig-to-keep-your-coins-ultra-safe/
129)
"Bitcoin's Lightning Network Payment Channel Explained !!", Sudhir Khatwani, 11 March 2019, https://themoneymongers.com/payment-channels/

2.3.3.4 Permanode

return to Node Taxonomy

Permanode is an IOTA-unique term for an authority node, a variant of an archival node that retains transaction data even after a snapshot is taken. It is not shown in the node taxonomy figure in Section 2.3.3 Node Taxonomy.

2.3.4 Data Taxonomy

return to Taxonomic Views

The third word in the acronym DIDO is *Data* and every node therefore, manages and controls data. The data within the node is classified as either: ledger data, ancillary data or external data.

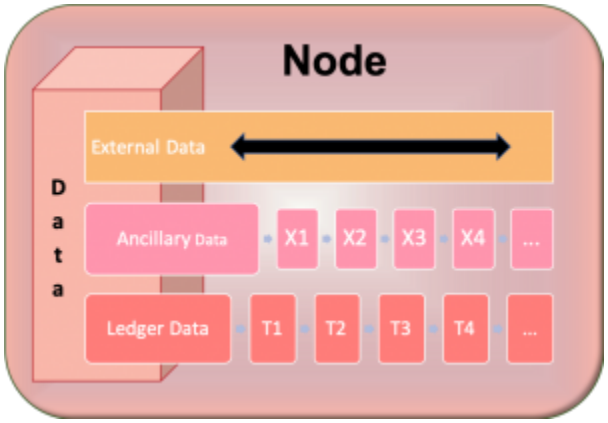


Figure 28: Types of Node Data

- Ledger Data
- Ancillary Data
- External Data

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

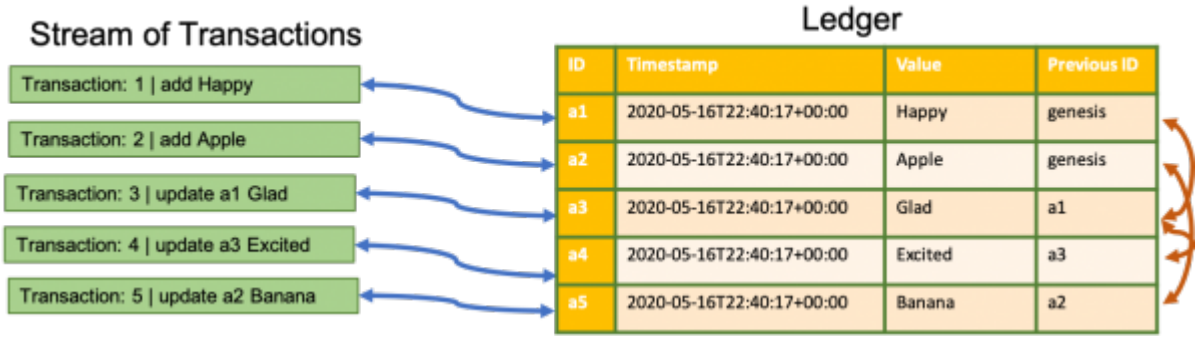
Tools

- None at this time

2.3.4.1 Ledger Data

return to Data Taxonomy

Ledger data is comprised of records containing immutable data. Even though the data captured within the ledger is immutable, this does not mean the ledger itself is immutable. The ledger can be updated by adding newer versions of existing data that then point back to the original data. Updates are made by applying transactions to the ledger in a prescribed order.



Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

2.3.4.2 Ancillary Data

return to Data Taxonomy

Ancillary data is adjunct information used in the formulation of transactions. It differs from transaction data because it is not used to define the operation described within a transaction. Probably the best way to explain ancillary data is using examples.

Examples of Ancillary Data

Supporting Data

> Data needs to be exchanged between two individuals that describe the exchange of one kind of currency with another (e.g., US dollars to EU euros or Bitcoins). The transaction can simply be something like this:

```
TRANSFER 500 USD to 1 Bitcoin in MyAccount
```

> The transaction seems legitimate and can probably be verified easily enough, or can it? In order to verify the transaction, the exchange rate used also needs to be provided. It might also require the timestamp of when the exchange rate was calculated and even where the exchange rate was obtained from. This “extra” information used to verify the transaction is called ancillary data. It ultimately needs to be coded into the transaction. A classic example of why ancillary data needs to be captured is to avoid salami slicing.

Business Process Management

> Data needs to be exchanged between two different business entities. Each business entity has its own business process describing the steps needed on its side in order to approve a transaction. The data required to approve the transaction is transformational in nature rather than transactional.

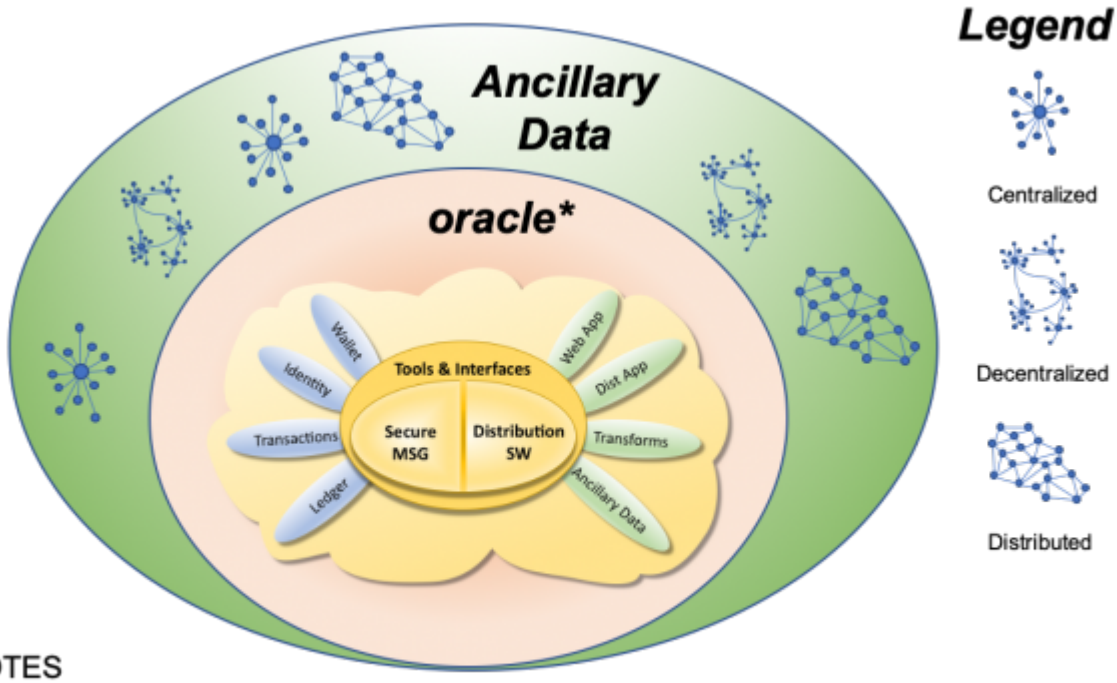
```
TRANSFER 200 AMZN Shares from MyOrg to YourOrg
```

> At a transactional level, all that is needed is verification that MyOrg has the shares, and YourOrg exists in order to prevent “double spend” fraud, i.e., spending the same money more than once. However, many organization have business processes in place that require signatures from various people and that these signatures must be in a prescribed sequence (i.e., Director signs, VP signs, Comptroller signs). To verify the transaction, the signatory data needs to be provided and

confirmed that it is valid. This can occur outside the transaction and be transformational in nature. Transformational data can be undone, deleted, or modified until the business process commits to the the transaction.

DIDO Implementation of Ancillary Data

Some DIDO implementations may provide direct support of ancillary data within the DIDO or they may provide access to external ancillary data through oracles. However, if ancillary data is not also implemented within the DIDO, but implemented externally through and accessed through the use of oracles, many of the benefits of the distributed architecture are lost because access to the ancillary data can become a bottleneck.



NOTES
* An oracle, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts. https://blockchainhub.net/blockchain-oracles/

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

2.3.4.3 External Data

return to Data Taxonomy

External data is data that is neither ledger data nor ancillary data but is required within the blockchain or smart contracts. The external data is accessed using an oracle. There are numerous formats for external data sources: data files, databases, rich site summary (RSS) feeds, RESTful interfaces, etc.

Standards

Technical Standards

- OMG: Data Distribution Service (DDS)
- RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format
- W3C: HTML5 (HTML5)
- Linux Foundation: Open Messaging
- Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)

de facto Standards

- None at this time

Tools

- None at this time

3 Governance

return to Reference Architecture

For practical reasons, the acronym Distributed Immutable Data Object (DIDO) represents a set of distributed computing technologies that focus on distributed data (i.e., blockchains, distributed ledgers, distributed file systems or distributed data). A major problem confronting the adoption of DIDO technologies is that it requires shifting away from corporate models of governance to an open community model of development, especially for industry or public ecosystems (e.g., financial, supply chains, public records) and domains (e.g., interest swaps, produce supply chain, cryptocurrencies, carbon credits, air pollution, traffic conditions). In the corporate model of governance, a single entity is responsible for the costs and the entire lifecycle of a product or project. Governance is accomplished through a formal chain of command, usually with a single individual responsible for the success or failure of the product or project.

However, in distributed computing, not all the resources are owned, paid for, or controlled by a single entity. In fact, the more entities involved in the distributed computing solution, the better. These differences in governance models make the adoption of distributed computing difficult by corporate entities since they have to rely on a larger, more inclusive community, which may include competitors, to measure the success of the solution. Ultimately, the success of the project or product comes down to controlling and minimizing risks. Being part of a larger, more diverse community increases some kinds of risks but may decrease others.

For example, the risks associated with specifying, architecting, designing, implementing, testing, maintaining, and sunsetting code are shared over the entire community, thus practically lowering the risks to any individual. The risks associated with setting requirements, priorities, and so on may be increased since these are now set externally to any single entity.

Currently, with the rise of open source solutions versus proprietary solutions, many efforts have moved from the centralized governing corporate model towards the decentralized, distributed community governing model with a great deal of success and broad adoption. Some examples of open source solutions are Operating System (OS), DataBase Management System (DBMS), application servers, web servers, file repositories, bug tracking tools, virtualization tools, and]]dido:public:ra:xapend.glossary:d:dlt]]. Most corporations would now rarely choose to build any of these products on their own, but have readily adapted to the adoption of these open source community products and solutions.

This section defines and recommends community of interest (CoI) governance structures needed for successful, robust, inclusive, and broadly adopted solutions. Some CoIs require a formal legal entity recognized by a government authority such as a state government. Other CoIs can operate within the confines of another legal CoI. For example, World Wide Web Consortium (W3C) and Object Management Group (OMG) are legal entities. Special interest groups (SIGs) or working groups can operate within the W3C or OMG.

Manage an Open Source Program

There have been many books written on this subject:[130)]

- RFC2026 - The Internet Standards Process
- TODO: Using open source code
- TODO: Participating in open source communities
- TODO: Recruiting open source developers
- TODO: Starting an open source project
- TODO: Improve your open source development impact
- TODO: Shutting down an open source project
- TODO: Building leadership in an open source community
- TODO: Setting an Open Source Strategy

130)

TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

3.1 DIDO Communities

Return to the Governance page

As introduced in Section 2.1, a DIDO community is a community of interest (CoI) responsible for the architecture, design, implementation, maintenance, and eventual sunset (yes, all good things come to an end) of DIDO networks and its DIDO nodes. DIDO communities should have a well-documented, repeatable, traceable, transparent, and unbiased process that is managed and run by a governing board (or hierarchy of governing boards). Often the DIDO community is divided into two organizations: one responsible for the software, usually as an open source software (OSS) project, and the other responsible for managing the fungible data (e.g., currency or commodity) managed by the software. Most DIDO communities choose or select DIDO software designed, implemented, and maintained by another party. For example, Ethereum, Hyperledger, and IOTA all provide their software for others to use as OSS.

Some DIDO communities may support multiple kinds of immutable data objects within a single DIDO network, or they can support multiple networks that each have different kinds of immutable data object types. For example, one DIDO community's OSS might support cryptocurrencies on one DIDO network and support immutable data objects representing supply chain commodities on another.

Unless a DIDO network's focus is very narrow, there is generally a need to have an exchange that allows different immutable data objects to be exchanged (e.g., US dollars to Bitcoins). The exchange might support the exchange of immutable data objects that are all contained within one DIDO network, or it might support the exchange of information managed by one DIDO community but on different DIDO networks. In this case, the exchange might manage converting customer loyalty reward points to a currency (i.e., euros or cryptocurrency) or supply chain commodities to cryptocurrencies and back. Other exchanges might exchange immutable data objects managed by one DIDO community and DIDO network with a cryptocurrency managed by another DIDO community; for example, the exchange of Bitcoins to Ethereum or Iotas.

3.1.1 Stakeholder Communities

Return to DIDO Communities Page

Given that DIDOs are centered around stakeholders, a stakeholder community is key to creating a thriving community of stakeholders. Unless the target of a DIDO is a private, permissioned network, there will be stakeholders that exist outside of any corporation and/or a corporation's customers or clients. In many cases, the stakeholder community will also include competitors or other participants (i.e., supply chain players) that want to integrate services around a common concept or idea. This common concept or idea is generally a distributed object.

In other words, the stakeholder community is generally defined as people, groups, organizations, or businesses that have interest or concern in the distributed object. Stakeholders affect or are affected by the community's actions, objectives and policies.

A stakeholder community can be informal or formal. Informal communities are best described as a confederation of the willing and generally have one or more core participants who make decisions for the group. When the stakeholder community is large, broad and with sometimes competing stakeholders, these communities should be formal, well organized organizations (usually non-profit).

Note: Refer to Section 2.1 Stakeholder Views for how the various stakeholder communities are organized, as well as the definitions and descriptions of ecosphere, ecosystem, and domain. Also, it should be understood that each of these terms refer to a kind of community of interest (CoI). Thus, saying “ecosphere” should be read as “ecosphere CoI”. Likewise for “domain” and “ecosystem”.

Steps for Establishing an Ecosphere

Steps for establishing and building a formal ecosphere CoI[131)]

1. Define a Mission Statement
2. Select a Leadership Team as Governing Board
3. Secure Funding
4. Create Legal Documents:
   a. Charter
   b. Bylaws
   c. Policy and Procedures (P&P)
5. Legally Incorporate
6. If Non-Profit, officially register as Non-Profit (In US, 501 ©(3))
7. Identify Partners (Founding Members)
8. Establish Milestones, Deliverables, and Schedules
9. Create Community Guides consistent with Charter, Bylaws, and Policies and Procedures
10. Recruit more members
11. Establish liaisons with other CoIs

131)

How To Start A Nonprofit Organization, Snowball Fundraising, Accessed 21 May 2020, https://snowballfundraising.com/starting-a-nonprofit-checklist/

3.1.2 Software Communities

Return to DIDO Communities Page

The cornerstone of a DIDO community is the software, which acts as the engine of the distributed network of peers. Software is responsible for maintaining the ledgers on each node: securely and reliably sending transactions to each of the nodes and mitigating any conflicts that may arise while processing these transactions (e.g., the double spend problem). As a general rule, the software is open source software (OSS), governed by OSS rules for its maintenance and the development of new features.

In some communities, the software extends beyond the core software required to maintain the node within a network of nodes. For example, the Ethereum Foundation includes both the software to maintain the network of nodes and smart contracts to monitor and oversee transactions. However, smart contracts themselves may or may not be built, maintained, and supported by the Ethereum Foundation's software team. They may be created, maintained, and supported by external third parties.

3.2 Legal Documents

return to Governance

☐ CHar Coule Review This Given the intentional distributed, decentralized nature of DIDOs, along with the potential for them to run on hardware and/or use software not owned or developed by a single entity, it is important to employ a community of interest (CoI) approach to develop, manage, and govern DIDO solutions. Such an approach allows for the establishment and specification of requirements for a formal organization that balance the various goals of its participants in pursuit of a single shared mission and, at the same time, protects the joint intellectual property (IP) resulting from their collaboration.

When an organization is a joint venture between different legal entities, it must operate in accordance with legal documents that serve to manage the expectations of, govern the interactions between, and elaborate the collective and individual deliverables expected from the legal entities participating in that organization. In the context of DIDO, *ecosphere* CoI is the term used to refer to such a joint venture. Table 2 lists the legal documents required to formally launch an ecosphere CoI. These documents support the process described in stakeholder communities. Depending on the country of incorporation, all three may be required for the ecosphere COI to be formally incorporated. In general terms (more details provided in Sections 3.2.1, 3.2.2, and 3.2.3), the charter defines how the ecosphere CoI interacts with the outside world; the bylaws define how it operates internally (e.g., fundamental governing rules, internal organization); policies & procedures (P&P), a document required for incorporation in the United States, establishes a set of principles to achieve the goals and vision of its charter and the specific methods employed to express these policies in action.

As already discussed in stakeholder views, other types of CoIs or special interest goups (SIGs) can be formed under the umbrella of a parent ecosphere: *ecosystems* and *domains*.[132)] As shown in Table 2, they must each be chartered according to the governing rules established in the bylaws and abide by the P&P of their parent ecosphere CoI. Ecosystem P&Ps may include specific extensions to the parent ecosphere's CoI P&P. In like manner, domain P&Ps may include specific extensions to the their parent ecosystem CoI P&P. All extensions to the parent P&P must be additive; they may not replace anything in the P&P of their parent ecosphere CoI.

Table 2: Documents required to Create and Govern a DIDO CoI

DIDO CoI	Charter	Bylaws	Policies and Procedures
Ecosphere	Yes(§)	Yes(†)	Yes(‡)
Ecosystem	Subcharter of Ecosphere	covered by Ecosphere	covered by Ecosphere + extensions
Domain	Subcharter of Ecosystem	covered by Ecosphere	covered by Ecosphere + extensions from Ecosystem _ local

- *((§) Initially, a legal statement created by the founders of the organization that lays out goals, missions and officers for the organization*
- *(†) a legal document reviewed by lawyers from all the participating parties*
- *(‡) Some Policies and Procedures may be mandated by law (i.e., discrimination, ADA, Safety, etc., others are local added by local governing boards) and should be drafted/reviewed by lawyers of all participating parties*

NOTE: There are no standards for these documents. The initial versions of these documents, particularly the Bylaws, should be drafted by attorneys. In the case of the P&P, the initial version should be drafted with input from attorneys to address legally mandated items.

- 3.2.1 Charter
- 3.2.2 Bylaws
- 3.2.3 Policies and Procedures (P&P)

132)

Described in more detail, along with illustrations, in Ecosystem View and Domain View.

3.2.1 Charter

return to Legal Documents

A charter is a legal document providing basic information about a community of interest (CoI): its location, purpose[133)], profit status (usually non-profit), governing board (e.g., board of directors) composition, and stakeholder (or ownership) structure. Sometimes the charter is referred to as the articles of incorporation or a certificate of incorporation. In the case of a for-profit corporation, the articles of incorporation must include the number, classes, and par values of authorized shares. In the United States, most states require the name and address of the company's registered agent as well. Source: Charters: Creating the Organization

The CoI charter is important because when it is filed and approved by a legal entity, such as a secretary of state, it "gives birth" to a new CoI as a formal corporation. A CoI charter is not the same as the CoI bylaws, which build off the charter and add things like fundamental governing rules, board meeting schedules, conditions of membership, etc.

Note:

1. Charter, as described in this section, only applies to ecosphere CoIs that file for incorporation.
2. Ecosystem CoIs and domain CoIs are also created using a charter, but these charters are not legal or legally binding documents. Instead they are "subcharters" of, and must be approved by, their parent ecosphere using a process defined in the parent ecosphere's P&P.

Essential Elements of a Charter

- **Name**
- **Address**
- **Statement of Purpose**
- **Profit Status (Usually non-Profit)**
- **Registered Agent**
- **Agent's Address**
- **Number of Shares Authorized**
- **The Classes and par value of the shares**
- **Roles** and **Responsibilities**
- **Directors**

Standards

- The charter ID made specifically for the CoI (i.e., ecosphere).

Tools

- Smartsheet Project Charter Template, Smartsheet, 21 May 2020, https://www.smartsheet.com/blog/project-charter-templates-and-guidelines-every-business-need
- UpBoard, Project Charter Online Tools & Templates – Best Practices, accessed 21 May 2020, https://upboard.io/project-charter-online-software-tools-templates/

References

- The Essential Guide to Creating an Effective Team Charter
- Investing Answers, Corporate Charter, Accessed 21 March 2020, https://investinganswers.com/dictionary/c/corporate-charter
- Investopedia, Understanding a Corporate Charter, Accessed 30 May 2020 https://www.investopedia.com/terms/c/corporatecharter.asp

133)

in the context of a DIDO CoI, think of this as another word for “Goal”, “Mission”, or “Vision”

3.2.2 Bylaws

return to Legal Documents

The community of interest (CoI) bylaws (or articles of organization) are the primary legal documents providing governance for the group. The bylaws are originally created during the initial steps of establishing the CoI (i.e., ecosphere). [134)]

Bylaws are supplemental to the rules already defined by the granting government entity's (e.g., states, counties, or cities in the US) code and rules for how the CoI must run. Bylaws specify the rules and regulations governing actions and decisions made by the board. The bylaws should help prevent or resolve conflicts and disagreements, and protect the CoI from potential problems by clearly outlining rules concerning authority levels, rights, and expectations.

> **Note**: If the board of directors fails to follow the adopted CoI bylaws, it can be held liable for breaching their duty to the CoI.

Common Provisions in Bylaws

1. Name and Purpose
2. Parliamentary Authority for Election, roles, and terms of officers and board members
3. Membership Issues (categories, responsibilities)
4. Meeting Guidelines (Frequency and Quorum)
5. Board Structure and Size
6. Compensation and indemnification of board members
7. Role of Chief Executive
8. Conflict of Interest Policy
9. Amendment of Bylaws Policy
10. Dissolution of the organization

Standards

- The bylaws are made specifically for the CoI (i.e., ecosphere).

de facto Standards

- Robert, Henry M.; et al. (2011). Robert's Rules of Order Newly Revised (11th ed.). Philadelphia, PA: Da Capo Press. ISBN 978-0-306-82021-2 (hardcover).
- Robert III, Henry M. (2011). “The Official Robert's Rules of Order Web Site (Home)”. The Official Robert's Rules of Order Web Site. The Robert's Rules Association.
- The Official Robert's Rules Of Order Web Site (robertsrules.com) Site maintained by the Robert's

Rules Association

Tools

- Section 7. Writing Bylaws, Community Tool Box, Accessed 21 May 2020, https://ctb.ku.edu/en/table-of-contents/structure/organizational-structure/write-bylaws/main
- Form of Bylaws - California Nonprofit Public Benefit Corporation, Public Coucil Org, Accessed 21 May 2020, http://www.publiccounsel.org/tools/publications/files/Annotated_Bylaws.pdf
- Sample Bylaws, U.S. Chamber of Commerce, Accessed 21 May 2020, https://www.uschamber.com/your-chamber-commerce-guide-starting-and-growing-chamber-commerce/sample-bylaws

References

- Bylaws for an Unincorporated Association, Drew Nelson, September 26, 2017, Accessed 21 May 2020, https://bizfluent.com/list-6981569-bylaws-unincorporated-association.html
- What is the difference between Charter and Bylaws?, William Adkins, bizfluet, 26 September 2017, Accessed 21 May 2020, https://bizfluent.com/facts-5894520-difference-between-charter-bylaws-.html
- The Difference Between Bylaws and Policy, Victoria Duff, 26 September 2017, Accessed 21 May 2020, https://bizfluent.com/facts-5921586-difference-between-bylaws-policy.html
- Bylaws, Policies and Procedures: What's the Difference?, Karen Blewett, presented to Board Leadership Southeast Alberta, 4 March 2017, Accessed 30 May 2020 https://boardleadershipsouth.com/files/march_4_2017/Bylaws,%20Policy%20and%20Procedures%202017.pdf

134)

Nonprofit Bylaws Made Easy: Tips and Best Practices, donorbox.org, Acccessed 21 May 2020, https://donorbox.org/nonprofit-blog/nonprofit-bylaws-made-easy/

3.2.3 Policies and Procedures (P&P)

return to Legal Documents

One of the legal documents required for the formal incorporation of an ecosphere community of interest (CoI) is the policies and procedures (P&P). The P&P establishes a set of principles and policies that serve these principles, in order to achieve the CoI's long-term goals (i.e, the purpose as stated in its charter). The P&P defines specific methods/procedures employed to express these policies in action: detailed rules and guidelines to control the activities of the organization. In the US, some policies are concerned with human resources (HR) issues (e.g., health and human safety, or data protection and security), a direct response to the General Data Protection Regulation (GDPR) or the Data Protection Act of 2018.

The intent of the P&P is to influence and help formulate all the major decisions and actions of the organization. All activities within the organization are to take place within the boundaries set by the P&P. [135)] Procedures are specific methods outlining the steps required to fulfill the policies on a day-to-day basis within the organization. Together, policies and procedures help the governing body of an organization meet its mission and goals.[136)]

One of the larger policies that is often required by the governing body granting incorporation are the human resources policies, especially those concerning health and human safety.

Standards

- The charter ID made specifically for the CoI (i.e., ecosphere).

Laws

- Occupational Safety and Health Act of 1970, Public Law 91-596, 84 STAT. 1590, 91st Congress, S.2193, December 29, 1970, as amended through January 1, 2004. https://www.osha.gov/laws-regs/oshact/completeoshact
- Regulation (EU) 2016/679 OF THE European Parliament and of the Council of 27 April 2016 https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

Tools

- Rocket Lawyer, Health and Safety Policy Document Maker, Accessed 21 may 2020, https://www.rocketlawyer.com/gb/en/documents/health-and-safety-policy
- Rocket Lawyer, Data Protection and Data Security Document Maker, Accessed 21 May 2020, https://www.rocketlawyer.com/gb/en/documents/data-protection-and-data-security-policy

References

- Health and Safety, Rocket Lawyer, accessed 21 May 2020. https://www.rocketlawyer.com/gb/en/quick-guides/health-and-safety
- Overview of the Data protection and data security policy, Rocket Lawyer, accessed 21 May 2020. https://www.rocketlawyer.com/gb/en/documents/data-protection-and-data-security-policy

135)

Difference Between Policies and Procedures, Key Differences, accessed 21 May 2020,https://keydifferences.com/difference-between-policies-and-procedures.html

136)

Policies and Procedures, Business Dictionary, http://www.businessdictionary.com/definition/policies-and-procedures.html

3.3 Guides

return to Governance

In addition to the legal documents governing the activities of a community of interest (CoI), many CoIs create a simple, easily understood guide to help its members maneuver through and understand all the governing statements contained in the statutes, charter, bylaws, parliamentary authority, special rules, custom practices, and standing rules associated with that particular CoI. For example, the OMG publishes a "Hitchhiker's Guide to the OMG Process". Even though it is not a “normative” document (has no legal standing whatsoever, cannot be cited as a defense for not following process), it is much easier to read and understand than the OMG Policies and Procedures. It provides additional details and insights that, even though they do not belong in a P&P, are essential to know. Technical standards bodies such as the IETF publish a similar guide, which is even more informal and flippant. In any case, the goal of such a guide is to make it easier for the members of a CoI to follow their CoI's process rules.

Appendix A: Glossary of Terms Related to DIDO

return to the Public Area **OR** return to Reference Architecture (RA)

The following terms are used by DIDO in various documents, web pages, etc.

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z				

A

Return to Glossary

Create a Glossary entry starting with 'A' **Word or Expression** → Add page

- Application
- Application Programming Interface
- Application Specific Integrated Circuit (ASIC)
- Assurance
- Authorization

Application

Return to Glossary

Application (more commonly known as an app) is software that bundles together certain features in a way that is accessible to a user. There are millions of apps on both the App Store and Android app stores, offering services (or verticals).

Source: Application

Application Programming Interface (API)

Return to Glossary

An **application programming interface (API)** is a set of protocols, routines, functions and/or commands that programmers use to develop software or facilitate interaction between distinct systems. APIs are available for both desktop and mobile use and are typically useful for programming Graphical User Interface (GUI) components, as well as allowing a software program to request and accommodate services from another program.

Source: Application Programming Interface (API)

Application Specific Integrated Circuit (ASIC)

Return to Glossary

Application Specific Integrated Circuit (ASIC) is basically an integrated circuit that is designed specifically for an individual purpose or application. Strictly speaking, this also implies that an ASIC is built for one and only one customer. The opposite of an ASIC is a standard product or general-purposed IC, such as a logic gate or a general-purposed microcontroller chip, both of which can be used across a large range of electronic applications. ASICs are usually classified into one of three categories; full custom, semi-custom, and structured

Source: Application Specific Integrated Circuit (ASIC)

Assurance

Return to Glossary

Assurance is the measure of confidence that the security features, practices, procedures, and architecture of an information system accurately mediates and enforces the security policy. - CNSS 4009 IA Glossary

Source: Committee on National Security Systems, CNS 4009 Glossary

Authorization

Return to Glossary

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. System administrators (SA) are typically assigned permission levels covering all system and user resources.

During authorization, a system verifies an authenticated user's access rules and either grants or refuses resource access.

Source: Authorization

B

Return to Glossary

Create a Glossary entry starting with 'B' **Word or Expression** → Add page

Bitcoin Wallet

Return to Glossary

A **Bitcoin wallet** is a software program where Bitcoins are stored. To be technically accurate, Bitcoins are not stored anywhere; there is a private key (secret number) for every Bitcoin address that is saved in the Bitcoin wallet of the person who owns the balance. Bitcoin wallets facilitate sending and receiving Bitcoins, and give ownership of the Bitcoin balance to the user. The Bitcoin wallet comes in many forms; desktop, mobile, web, and hardware are the four main types of wallets.

Source: Bitcoin Wallet

Block Producers

Return to Glossary

Block Producers in Delegated Proof of Stake (DPoS) are full nodes responsible for creating and signing new blocks. They are limited in number, and are elected by the voters.

Source: Block Producers

Block Validators

Return to Glossary

Block Validators in Delegated Proof of Stake (DPoS) refer to full nodes that verify the blocks created by Block Producers and follow consensus rules. Any user is able to run a block validator and verify the network. (This can be confusing, since in Casper's PoS, the word "validators" refers to those who create blocks).

Source: Block Validators

Blockchain

Return to Glossary

Blockchain is a critical part of the bitcoin peer-to-peer payment system. The bitcoin system works using a blockchain ledger to record transactions. Bitcoin is a global cryptocurrency that can be used as a medium of exchange. However, while many parties have started to accept bitcoin as a currency, it is still controversial and poses risks in terms of security and stability.

Source: Blockchain

Blockchain Network

Return to Glossary

A **Blockchain_network** is a technical infrastructure that provides ledger and smart contract (chaincode) services to applications. Primarily, smart contracts are used to generate transactions which are subsequently distributed to every peer node in the network where they are immutably recorded on their copy of the ledger. The users of applications might be end users using client applications or blockchain network administrators.

Source: Blockchain Network

Brownfield

Return to Glossary

Brownfield refers to the implementation of new systems to resolve IT problem areas while accounting for established systems. New software architecture must account for existing and running software.

A commonly used IT term, Brownfield was borrowed from the building industry, where brownfield land describes a geographical location where new buildings may be constructed after considering the area's established structures and services.

Source: Brownfield

Bylaws

Return to Glossary

Bylaws are legal documents that establish the internal structure and governing rules of the organization. Bylaws provide the framework for internal governance and day-to-day operations of its governing structure.

Source: Bylaws

Byzantine Fault Tolerance

Return to Glossary

Byzantine Fault Tolerance is the characteristic of a system that is able to tolerate a class of failures called the Byzantine Generals' Problem[137)].

137)

"Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance", Georgios Konstantopoulos, 1 December 2017, Byzantine Fault Tolerance

Byzantine Generals Problem

Return to Glossary

Byzantine Generals Problem is when a reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked – namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. [138)]

138)

The Byzantine Generals Problem“, Leslie Lamport, Robert Shostak, and Marshal Pesse, SRI International, July 1983, Byzantine Generals Problem

C

Return to Glossary

Create a Glossary entry starting with 'C' **Word or Expression** →		Add page

Central Processing Unit (CPU)

Return to Glossary

Central Processing Unit (CPU) is the primary component of a computer that processes instructions. It runs the operating system and applications, constantly receiving input from the user or active software programs. It processes the data and produces output, which may stored by an application or displayed on the screen.

Source: Central Processing Unit (CPU)

Charter

Return to Glossary

A **Charter** is a legal document that creates for-profit or nonprofit organizations. Frequently called articles of incorporation, a charter brings the organization into existence as a legal entity. Charters must be filed with an approval authority such as the secretary of state for the state where the organization is located.

Source: Charter

Coins

Return to Glossary

Coins (also often called *altcoins* or *alternative cryptocurrency coins*) are digital money, created using encryption techniques, that store value over time. Basically they are the digital equivalent of money. Bitcoin is the most famous example. Bitcoin is based on blockchain — public and distributed digital ledger — where all transactions can be seen. Data is stored collectively and shared between participants of a blockchain network. Blockchain guarantees transparency and reduces fraud.

Coins have the same characteristics as money: they are fungible, divisible, acceptable, portable, durable and have limited supply. Most ambitious crypto enthusiasts insist that coins will replace conventional money in the future.

The main characteristics of coins are:

1. they are tied to public-open blockchain—anyone is allowed to join and participate in the network;
2. they may be sent, received or mined.

Coins are not meant to perform any functions beyond acting as money.

Compare to tokens

Source: Coins

Command Line Interface (CLI)

Return to Glossary

Command Line Interface (CLI) is is a text-based interface that is used to operate software and operating systems while allowing the user to respond to visual prompts by typing single commands into the interface and receiving a reply in the same way.

CLI is quite different from the Graphical User Interface (GUI) that is presently being used in the latest operating systems.

Source: Command Line Interface (CLI)

Common Intermediate Language (CIL)

Return to Glossary

Common Intermediate Language (CIL), formerly called Microsoft Intermediate Language (MSIL) or Intermediate Language (IL),[1] is the intermediate language binary instruction set defined within the Common Language Infrastructure (CLI) specification.[2] CIL instructions are executed by a CLI-compatible runtime environment such as the Common Language Runtime. Languages which target the CLI compile to CIL. CIL is object-oriented, stack-based bytecode. Runtimes typically just-in-time compile CIL instructions into native code.

Source: Common Intermediate Language (CIL)

Common Language Runtime (CLR)

Return to Glossary

Common Language Runtime (CLR) is is a managed execution environment that is part of Microsoft's .NET framework. CLR manages the execution of programs written in different supported languages.

CLR transforms source code into a form of bytecode known as Common Intermediate Language (CIL). At run time, CLR handles the execution of the CIL code.

Source: Common Language Runtime (CLR)

Communication Protocol

Return to Glossary

Communication protocols are formal descriptions of digital message formats and rules. They are required to exchange messages in or between computing systems and are required in telecommunications. Communication protocols cover authentication, error detection and correction, and signaling. They can also describe the syntax, semantics, and synchronization of analog and digital communications.

Source: Communication Protocol

Community of Interest (CoI)

Return to Glossary

Community of Interest (CoI) is a collaborative group of users who exchange information in pursuit of their shared goals, interests, missions, or business processes, and who therefore must have a shared vocabulary for the information they exchange. The group exchanges information within and between systems to include security domains.

Source: Community of Interest

Configuration Management (CM)

Return to Glossary

Configuration Management (CM) is a process for maintaining computer systems, servers, and software in a desired, consistent state. It's a way to make sure that a system performs as expected as changes are made over time.

Source: Configuration Management (CM)

Consensus Algorithm

Return to Glossary

A **Consensus Algorithm** is a process in computer science used to achieve agreement on a single data value among distributed processes or systems. Consensus algorithms are designed to achieve reliability in a network involving multiple unreliable nodes. Solving that issue – known as the consensus problem – is important in distributed computing and multi-agent systems.

Source: Consensus Algorithm - Tech Target, April 2018

Copyleft

Return to Glossary

Copyleft refers to licenses that allow derivative works but require them to use the same license as the original work. For example, if you write some software and release it under the GNU General Public License (a widely-used copyleft license), and then someone else modifies that software and distributes their modified version, the modified version must be licensed under the GNU GPL too — including any new code written specifically to go into the modified version. Both the original and the new work are Open Source; the copyleft license simply ensures that property is perpetuated to all downstream derivatives. (There is at least one copyleft license, the Affero GPL, that even requires you to offer the source code, under the AGPL, to anyone to whom you make the software's functionality available as a network service — however, most copyleft licenses activate their share-and-share-alike requirement on distribution of a copy of the software itself. You should read the license to understand its requirements for source code distribution.)

Most copyleft licenses are Open Source, but not all Open Source licenses are copyleft. When an Open Source license is not copyleft, that means software released under that license can be used as part of programs distributed under other licenses, including proprietary (non-open-source) licenses. For example, the BSD license is a non-copyleft Open Source license. Such licenses are usually called either “non-copyleft” or “permissive” open source licenses

Copyleft provisions apply only to actual derivatives, that is, cases where an existing copylefted work was modified. Merely distributing a copyleft work alongside a non-copyleft work does not cause the latter to fall under the copyleft terms.

Source: What is "copyleft"? Is it the same as "open source"?

D

Return to Glossary

Create a Glossary entry starting with 'D' **Word or Expression** → Add page

- Data as a Service (DaaS)
- DataBase Management System (DBMS)
- Data Distribution Service (DDS)
- Data Model (DM)
- Data Protection
- Datastore
- de facto Standard
- Delegated Byzantine Fault Tolerant (dBFT)
- Delegated Proof of Stake (DPoS)
- Department of Defense (DoD)
- Directed Acyclic Graph (DAG)
- Disconnected, Intermittent and Limited (DIL)
- Distributed Application (ĐApp or DApp)
- Distributed Immutable Data Objects (DIDO)
- Distributed Ledger Technology (DLT)
- Domain Name System (DNS)

Data as a Service (DaaS)

Return to Glossary

Data as a service (DaaS) is a cloud strategy used to facilitate the accessibility of business-critical data in a well-timed, protected and affordable manner. DaaS depends on the principle that specified, useful data can be supplied to users on demand, irrespective of any organizational or geographical separation between consumers and providers.

DaaS eliminates redundancy and reduces associated expenditures by accommodating vital data in a single location, allowing data use and/or modification by multiple users via a single update point. Initially used in Web mashups, the DaaS strategy is often used by commercial organizations.

Source: Data as a Service (DaaS)

DataBase Management System (DBMS)

Return to Glossary

A DataBase Management System (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

A DBMS relieves users of framing programs for data maintenance. Fourth-generation query languages, such as SQL, are used along with the DBMS package to interact with a database.

Source: DataBase Management System (DBMS)

Data Distribution Service (DDS)

Return to Glossary

The Data Distribution Service (DDS™) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group (OMG). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical Internet of Things (IoT) applications need. Source: Data Distribution Service (DDS)

Data Model (DM)

Return to Glossary

A **Data Model (DM)** refers to the logical inter-relationships and data flow between different data elements involved in the information world. It also documents the way data is stored and retrieved. Data models facilitate communication business and technical development by accurately representing the requirements of the information system and by designing the responses needed for those requirements. Data models help represent what data is required and what format is to be used for different business processes.

Source: Data Model (DM)

Data Protection

Return to Glossary

Data Protection is the process of safeguarding important information from corruption, compromise or loss.

Source: Data Protection

Datastore

Return to Glossary

A **Datastore** is a repository for storing, managing and distributing data sets on an enterprise level. It is a broad term, which incorporates all types of data that is produced, stored and used by an organization. The term refers specifically to data that is at rest and used by one or more data-driven applications, services or individuals. A database is a kind of **datastore**, but there are datastores that are not databases, for example W3C documents or JSON files.

Source: Datastore

de facto Standard

Return to Glossary

A de facto standard is something that is used so widely that it is considered a standard for a given application although it has no official status.

Source: De facto Standard

Delegated Byzantine Fault Tolerant (dBFT)

Return to Glossary

Delegated Byzantine Fault Tolerant (dBFT) is a consensus mechanism that was made popular by a cryptocurrency called NEO. dBFT essentially works in a similar fashion to a country's governance system, having its own citizens, delegates, and speakers to ensure that the country (Node Network) is functional. The method is closer to Proof of Stake (PoS) rather than Proof of Work (PoW), by utilizing a voting system to choose delegates and speaker.

Source: Delegated Byzantine Fault Tolerant (dBFT)

Delegated Proof of Stake (DPoS)

Return to Glossary

Delegated Proof of Stake (DPoS) is an alternative consensus algorithm to the Proof of Work (PoW) or Proof of Stake (PoS) algorithms. DPoS is a system in which a fixed number of elected entities (called Block Producers or witnesses) are Block Validators selected to create blocks in a round-robin order. Block Producers are voted into power by the users of the Node Network, that each get a number of votes proportional to the number of tokens they own on the network (their stake).

Alternatively, voters can choose to delegate their stake to another voter, who will vote in the Block Producer election on their behalf.

> **Note:** DPoS is a protocol that sacrifices decentralization for throughput due to the low number of Block Producers.

Source: Delegated Proof of Stake DPoS

Department of Defense (DoD)

Return to Glossary

The United States **Department of Defense (DoD)** is the department within the United States that provides the military forces needed to deter war, and to protect the security of the United States. Source: Department of Defense (DoD)

Directed Acyclic Graph (DAG)

Return to Glossary

Directed Acyclic Graph (DAG) mathematically a DAG is a graph that travels in one direction without cycles connecting the other edges. This means it is impossible to traverse the entire graph starting at one edge. The edges of the directed graph only go one way. The graph is a topological sorting, where each node is in a certain order.

Imagine a collection of individual transactions where each transaction is linked to at least one other transaction in the following way:

Directed

The links point in the same direction with earlier transactions linked to later transactions, and so on.

Acyclic

Loops are not possible. A transaction cannot loop back on itself after linking to another transaction.

Graph

The mesh of connected transactions can be represented as nodes in a graph network, in which nodes are joined to each other by links.[139)]

139)
“What is DAG Distributed Ledger Technology?”, Max Thake, 9 November 2018, https://medium.com/nakamo-to/what-is-dag-distributed-ledger-technology-8b182a858e19

Disconnected, Intermittent and Limited (DIL)

Return to Glossary

Disconnected, Intermittent and Limited (DIL) occur in wireless/mobile networking environments, e.g. rural area networks, vehicular networks, battlefield networks and other resource-constrained or disadvantaged networks. Efficient and effective interoperability in these networks is highly demanded for various mission-critical application scenarios such as disaster relief in ravaged regions, search and rescue in remote areas and military/tactical operations in hostile environments.

Source: Disconnected, Intermittent and Limited (DIL)

Distributed Application (ĐApp or DApp)

Return to Glossary

Distributed Application (ĐApp or DApp) is software that is executed or run on multiple computers within a network. These applications interact in order to achieve a specific goal or task. Traditional applications relied on a single system to run them. Even in the client-server model, the application software had to run on either the client, or the server that the client was accessing. However, distributed applications run on both simultaneously. With distributed applications, if a node that is running a particular application goes down, another node can resume the task.

Source: Distributed Application

Distributed Immutable Data Objects (DIDO)

Return to Glossary

The term Distributed Immutable Data Objects (DIDO) refers to the underlying technologies supporting distributed data and computation across a distributed network of peers using consensus algorithms to maintain integrity and consistency across the network.

Source: Distributed Immutable Data Objects (DIDO)

Distributed Ledger Technology (DLT)

Return to Glossary

Distributed Ledger Technology (DLT) is a digital system for recording the transaction of assets in which the transactions and their details are recorded in multiple places at the same time. Unlike traditional databases, distributed ledgers have no central data store or administration functionality.

Source: Distributed Ledger Technology (DLT)

Domain Name System (DNS)

Return to Glossary

Domain Name System (DNS) is a hierarchical naming system built on a distributed database. This system transforms domain names to IP addresses and makes it possible to assign domain names to groups of Internet resources and users, regardless of the entities' physical location.

Source: Domain Name System (DNS)

E

Return to Glossary

Create a Glossary entry starting with 'E' **Word or Expression** →		Add page

- Elastic Compute Cloud (EC2)
- Endianness
- Ethereum Improvement Proposal (EIP)
- Ethereum Request for Comment (ERC)

Elastic Compute Cloud (EC2)

Return to Glossary

Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Source: Elastic Compute Cloud (EC2)

Endianness

Return to Glossary

endianness is the byte order chosen for all digital computing made in a specific computer system and dictates the architecture and low-level programming approach to be used for that system.

Source: Endian

Ethereum Improvement Proposal (EIP)

Return to Glossary

Ethereum Improvement Proposal (EIP) is a design document providing information to the Ethereum community, or describing a new feature for Ethereum or its processes or environment. The EIP should provide a concise technical specification of the feature and a rationale for the feature. The EIP author is responsible for building consensus within the community and documenting dissenting opinions.

Source: Ethereum Improvement Proposal (EIP)

Ethereum Request for Comment (ERC)

Return to Glossary

Ethereum Request for Comment (ERC) is a technical document used by smart contract developers at Ethereum. They define a set of rules required to implement tokens for the Ethereum ecosystem. These documents are usually created by developers, and they include information about protocol specifications and contract descriptions. Before becoming an standard, an ERC must be revised, commented and accepted by the community through an EIP (Ethereum Improvement Proposal).

Source: Ethereum Request for Comment (ERC)

F

Return to Glossary

Create a Glossary entry starting with 'F' **Word or Expression →** Add page

- Fifty-One Percent (51% Attack)
- Financial Instrument Global Identifier (FIGI)
- FIGI Symbology
- Full Node
- Fungible

Fifty-One Percent (51% Attack)

Return to Glossary

Fifty-One Percent (51% Attack) refers to an attack on a blockchain – usually bitcoin's, for which such an attack is still hypothetical – by a group of miners controlling more than 50% of the network's mining hashrate, or computing power. The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between some or all users. They would also be able to reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins.

They would almost certainly not be able to create a create new coins or alter old blocks, so a 51% attack would probably not destroy bitcoin or another blockchain-based currency outright, even if it proved highly damaging.

Source: Fifty-One Percent (51% Attack)

Financial Instrument Global Identifier (FIGI)

Return to Glossary

The **Financial Instrument Global Identifier (FIGI)** is (formerly Bloomberg Global Identifier (BBGID)) is an open standard, unique identifier of financial instruments that can be assigned to instruments including common stock, options, derivatives, futures, corporate and government bonds, municipals, currencies, and mortgage products

Source: Financial Instrument Global Identifier

FIGI Symbology

Return to Glossary

FIGI symbology consists of the unique, persistent, unchanging alpha-numeric identifier, as well as the multiple individual pieces of associated descriptive metadata.

Source: Standard Symbology for Global Financial Securities

Full Node

Return to Glossary

A **Full Node** is a complete copy of the blockchain and is able to verify all transactions since the beginning. This requires a lot of disk space.

Source: Full Node

Fungible

Return to Glossary

Fungible (of goods contracted for without an individual specimen being specified) replaceable by another identical item; mutually interchangeable. Example: ‘it is by no means the world’s only fungible commodity’

Source: fungible

G

Return to Glossary

Create a Glossary entry starting with 'G' **Word or Expression** → Add page

- General Data Protection Regulation (GDPR)
- Google Mobile Services (GMS)
- Graphical User Interface (GUI)
- Graphics Processing Unit (GPU)
- Greenfield

General Data Protection Regulation (GDPR)

Return to Glossary

General Data Protection Regulation (GDPR) is a legal framework that sets guidelines for the collection and processing of personal information from individuals who live in the European Union (EU). Since the Regulation applies regardless of where websites are based, it must be heeded by all sites that attract European visitors, even if they don't specifically market goods or services to EU residents.

The GDPR mandates that EU visitors be given a number of data disclosures. The site must also take steps to facilitate such EU consumer rights as a timely notification in the event of personal data being breached. Adopted in April 2016, the Regulation came into full effect in May 2018, after a two-year transition period. Source: General Data Protection Regulation (GDPR)

Google Mobile Services (GMS)

Return to Glossary

Google Mobile Services (GMS) are the apps by Google that often come pre-installed on Android devices. GMS is not a part of the Android Open Source Project (AOSP), which means an Android manufacturer needs to obtain a license from Google in order to legally pre-install GMS on an Android device. This license is provided by Google without any license fees.

GMS consists of two parts; a popular bundle package and an other bundle package. In order to gain a license for GMS, the popular bundle package needs to be pre-installed by Android device manufactures, which are usually called pre-loaded apps.

Source: Google Mobile Services (GMS)

Graphical User Interface (GUI)

Return to Glossary

Graphical User Interface (GUI) is a type of user interface through which users interact with electronic devices via visual indicator representations.

Source: Graphical User Interface (GUI)

Graphics Processing Unit (GPU)

Return to Glossary

Graphics Processing Unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing. Their highly parallel structure makes them more efficient than general-purpose Central Processing Unit (CPU) for algorithms that process large blocks of data in parallel. In a personal computer, a GPU can be present on a video card or embedded on the motherboard. In certain CPUs, they are embedded on the CPU die.

Source: Graphics Processing Unit (GPU)

Greenfield

Return to Glossary

Greenfield is a type of deployment that refers to the installation of an IT system where previously there was none. This term is derived from the construction industry, where new development on previously undeveloped land is called greenfield development. Greenfield deployment may refer to a network, data center or other major IT projects when they are built from the ground up. This type of development is often beneficial because it is not subject to constraints posed by existing networks.

Greenfield networks may also be referred to as greenfield projects.

Source: Greenfield

H

Return to Glossary

Create a Glossary entry starting with 'H' **Word or Expression** → Add page

- Hard Fork
- Health Insurance Portability and Accountability Act (HIPAA)
- Hybrid Network
- Hype Cycle

Hard Fork

Return to Glossary

A **hard fork** (or sometimes **hardfork**), as it relates to blockchain technology, is a radical change to the protocol that makes previously invalid blocks/transactions valid (or vice-versa). This requires all nodes or users to upgrade to the latest version of the protocol software.

Source: Hard Fork

Health Insurance Portability and Accountability Act (HIPAA)

Return to Glossary

Health Insurance Portability and Accountability Act (HIPAA) a US law designed to provide privacy standards to protect patients' medical records and other health information provided to health plans, doctors, hospitals and other health care providers. Developed by the Department of Health and Human Services, these new standards provide patients with access to their medical records and more control over how their personal health information is used and disclosed. They represent a uniform, federal floor of privacy protections for consumers across the country. State laws providing additional protections to consumers are not affected by this new rule. HIPAA took effect on April 14, 2003.

Source: Health Insurance Portability and Accountability Act

Hybrid Network

Return to Glossary

Hybrid Blockchain is unique in that it is decentralized while also making it possible to restrict the visibility of information on the network with a combination of Public, Private, Permissionless and Permissioned Networks. Thus, a hybrid blockchain is appealing for regulated markets as it offers the benefits of public blockchain and private blockchain together.[140)]

140)
"Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.", Mina Down, 14 November 2018, Source: https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

Hype-Cycle

Return to Glossary

The **hype-cycle** is a graphical representation of the life cycle stages a technology goes through from conception to maturity and widespread adoption.

Source: Gartner hype cycle

I

Return to Glossary

Create a Glossary entry starting with 'I' **Word or Expression** → Add page

Identification

Return to Glossary

Identification is the starting point for all access control as without proper identification it will not be possible to grant resources to any identity. The main objective of identification is to bind a user to appropriate controls based on the identity.

Source: Identification

Immutable

Return to Glossary

Immutable is an unchanging over time or unable to be changed[141]. The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. [142]

141)
"Immutable", Lexico, Oxford English Dictionary, 28 June 2018,
142)
"Blockchain Technology Explained: Introduction, Meaning, and Applications", Mayank Pratap, 1 August 2018, Hackernoon, https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2

Industrial Internet of Things (IIoT)

Return to Glossary

Industrial Internet of Things (IIoT) is a term for all of the various sets of hardware pieces that work together through internet of things connectivity to help enhance manufacturing and industrial processes. When people talk about the industrial internet of things, they're talking about all of the sensors, devices and machines that contribute to physical business processes in industrial settings. By contrast, when people talk about the internet of things in general, they're talking about any connected devices that fit the IoT model – for instance, when people think about the internet of things, they often think about smart home devices that are linked together to provide consumer conveniences. Source: Industrial Internet of Things (IIoT)

Information Assurance (IA)

Return to Glossary

Information Assurance (IA) are measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. These measures include providing for restoration of information systems by incorporating protection, detection, and reaction capabilities

Source: Committee on National Security Systems, CNS 4009 Glossary

Source: NYSE Assurance

Information Security (IS/InfoSec)

Return to Glossary

Information security, often referred to as **InfoSec**, is designed to protect the confidentiality, integrity and availability of computer system data from those with malicious intentions. Confidentiality, integrity and availability are sometimes referred to as the CIA Triad of information security. This triad has evolved into what is commonly termed the Parkerian hexad, which includes confidentiality, possession (or control), integrity, authenticity, availability and utility

Source: Information Security (IS/InfoSec)

Information Technology (IT)

Return to Glossary

Information Technology (IT) is a business sector that deals with computing, including hardware, software, telecommunications and generally anything involved in the transmittal of information or the systems that faciliate communication.

Source: [[https://www.techopedia.com/definition/626/information-technology-it | Information Technology (IT)

Infrastructure-as-a-Service (IaaS)

Return to Glossary

Iaas is ...Infrastructure-as-a-Service (IaaS), also known as cloud infrastructure services, is a form of cloud computing in which IT infrastructure is provided to end users through the internet. IaaS is commonly associated with serverless computing.

Source: Infrastructure-as-a-Service (IaaS)

Initial Coin Offering (ICO)

Return to Glossary

Initial Coin Offering (ICO) is the cryptocurrency industry's equivalent to an Initial Public Offering (IPO). ICOs act as a way to raise funds, where a company looking to raise money to create a new coin, app, or service launches an ICO. Interested investors can buy into the offering and receive a new cryptocurrency token issued by the company. This token may have some utility in using the product or service the company is offering, or it may just represent a stake in the company or project.

Source: Initial Coin Offering (ICO)

Intellectual Property (IP)

Return to Glossary

Intellectual Property (IP) is any product of the human intellect that the law protects from unauthorized use by others. The ownership of intellectual property inherently creates a limited monopoly in the protected property. Intellectual property is traditionally comprised of four categories: patent, copyright, trademark, and trade secrets. Source: Intellectual Property (IP)

Interface

Return to Glossary

Interface is a boundary across which two independent systems meet and act on or communicate with each other. In computer technology, there are several types of interfaces.

Source: Interface

Internet of Things (IOT)

Return to Glossary

Internet of Things (IOT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1

Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2

From a broader perspective, the IoT can be perceived as a vision with technological and societal

implications.

Source: ITU-T Y.2060 - Overview of the Internet of things

Internet Protocol (IP)

Return to Glossary

Internet Protocol (IP) is the principal set (or communications protocol) of digital message formats and rules for exchanging messages between computers across a single network or a series of interconnected networks, using the Internet Protocol Suite (often referred to as TCP/IP). Messages are exchanged as datagrams, also known as data packets or just packets.

IP is the primary protocol in the Internet Layer of the Internet Protocol Suite, which is a set of communications protocols consisting of four abstraction layers: link layer (lowest), Internet layer, transport layer and application layer (highest).

The main purpose and task of IP is the delivery of datagrams from the source host (source computer) to the destination host (receiving computer) based on their addresses. To achieve this, IP includes methods and structures for putting tags (address information, which is part of metadata) within datagrams.

The process of putting these tags on datagrams is called encapsulation.

Source: Internet Protocol (IP)

J

Return to Glossary

Create a Glossary entry starting with 'J' **Word or Expression** →		Add page

- Just-In-Time (JIT)

Just-In-Time (JIT)

Return to Glossary

Just-In-Time (JIT) compiler is a an essential part of the JRE i.e. Java Runtime Environment, that is responsible for performance optimization of java based applications at run time. Compiler is one of the key aspects in deciding performance of an application for both parties i.e. the end user and the application developer.

Source: Just-In-Time (JIT)

K

Return to Glossary

Create a Glossary entry starting with 'D' **Word or Expression** →	Add page

- Know Your Customer (KYC)

Know Your Customer (KYC)

Return to Glossary

Know Your Client or Know Your Customer (KYC) is a standard in the investment industry that ensures investment advisors know detailed information about their clients' risk tolerance, investment knowledge, and financial position. KYC protects both clients and investment advisors. Clients are protected by having their investment advisor know what investments best suit their personal situations. Investment advisors are protected by knowing what they can and cannot include in their client's portfolio. KYC compliance typically involves requirements and policies such as risk management, customer acceptance policies, and transaction monitoring. Source: Know Your Customer (KYC)

L

Return to Glossary

Create a Glossary entry starting with 'L' **Word or Expression** → Add page

- Ledger
- License Distribution
- License Linking
- License Modification
- License Patent Grant
- License Private Use
- Licensing Sublicensing
- Licensing Trademark Grant
- Lightning Network
- Light Node

Ledger

Return to Glossary

Ledger – is a collection of an entire group of similar accounts in double-entry bookkeeping. Also, called book of final entry, a ledger records classified and summarized financial information from journals (the 'books of first entry') as debits and credits, and shows their current balances. In manual accounting systems, a ledger is usually a loose-leaf binder with a separate page for each ledger account. In computerized systems, it consists of interlinked digital files, but follows the same accounting principles as the manual system.

Source: Ledger

License Distribution

Return to Glossary

License Distribution is distribution of the code to third parties.

Source: Distribution

License Linking

Return to Glossary

License Linking is - linking of the licensed code with code licensed under a different license (e.g. when the code is provided as a library)

Source: Linking

License Modification

Return to Glossary

License Modification is modification of the code by a licensee.

Source: Modification

License Patent Grant

Return to Glossary

License Patent Grant is protection of licensees from patent claims made by code contributors regarding their contribution, and protection of contributors from patent claims made by licensees.

Source: Patent Grant

License Private Use

Return to Glossary

License Private Use is whether modification to the code must be shared with the community or may be used privately (e.g. internal use by a corporation)

Source: Private Use

Licensing Sublicensing

Return to Glossary

Licensing Sublicensing is whether modified code may be licensed under a different license (for example a copyright) or must retain the same license under which it was provided.

Source: Sublicensing

Licensing Trademark Grant

Return to Glossary

Licensing Trademark Grant is use of trademarks associated with the licensed code or its contributors by a licensee.

Source: Trademark Grant

Lightning Network

Return to Glossary

A **Lightning Network** is network layered on top of an existing blockchain technology (i.e., Bitcoin) intended to help with scalability issues. It is a decentralized system for instant, high-volume micropayments that removes the risk of delegating custody of funds to trusted third parties using a peer-to-peer (P2P) network connections called Micropayment Channel or Payment Channel. [143)]

The Lightning Network solves these problems. It is one of the first implementations of a multi-party Smart Contract (programmable money) using bitcoin's built-in scripting. The Lightning Network is leading technological development in multiparty financial computations with bitcoin.

Instant Payments:

Bitcoin aggregates transactions into blocks spaced ten minutes apart. Payments are widely regarded as secure on bitcoin after confirmation of six blocks, or about one hour. On the Lightning Network, payments don't need block confirmations, and are instant and atomic. Lightning can be used at retail point-of-sale terminals, with user device to-device transactions, or anywhere instant payments are needed.

Micropayments:

New markets can be opened with the possibility of micropayments. Lightning enables one to send funds down to 0.00000001 bitcoin without custodial risk. The bitcoin blockchain currently enforces a minimum output size many hundreds of times higher, and a fixed per-transaction fee which makes micropayments impractical. Lightning allows minimal payments denominated in bitcoin, using actual bitcoin transactions.

Scalability:

The bitcoin network will need to support orders of magnitude higher transaction volume to meet demand from automated payments. The coming increase in internet-connected devices needs a platform for machine-to-machine payments and automated micropayment services. Lightning Network transactions are conducted to the blockchain without delegation of trust and ownership, allowing users to conduct nearly unlimited transactions between other devices.

How it Works?

Funds are placed into a two-party, multisignature “channel” bitcoin address. This channel is

represented as an entry on the bitcoin public ledger. In order to spend funds from the channel, both parties must agree on the new balance. The current balance is stored as the most recent transaction signed by both parties, spending from the channel address. To make a payment, both parties sign a new exit transaction spending from the channel address. All old exit transactions are invalidated by doing so.

The Lightning Network does not require cooperation from the counterparty to exit the channel. Both parties have the option to unilaterally close the channel, ending their relationship. Since all parties have multiple multisignature channels with many different users on this network, one can send a payment to any other party across this network.

143)

“The Bitcoin Lightning Network”, Lightning Network

Light Node

Return to Glossary

A **Light Node** does not store the Tangle and you don't need any neigbours. You just need to select a host (e.g. `https://node.tangle.works:443`) which is a full node itself. This host provides your wallet with all the necessary information to make transaction.

Source: Light Node

M

Return to Glossary

Create a Glossary entry starting with 'M' **Word or Expression →** Add page

- Maintainability Measure
- Message Queue(MQ)
- Micropayment Channel
- Miner Node
- Mission Assurance (MA)
- Multi-Signature (multisig)

Maintainability Measure

Return to Glossary

Maintainability represents the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers (ISO 25010). Maintainability incorporates such concepts as changeability, modularity, understandability, testability, and reusability. Maintainability is responding rapidly to market conditions and keeping IT costs under control. The Maintainability of an application is a combination of compliance with good coding practices, the homogeneity with which coding rules are applied across an application, and compliance with architectural rules.

Source: IETF - The OAuth 2.0 Authorization Framework - October 2012

Message Queue(MQ)

Return to Glossary

Message Queue(MQ) is a software engineering component used for communication between processes or between threads within the same process. ... Message queues are used within operating systems or applications as a way for programs to communicate with one another. Source: Message Queue(MQ)

Micropayment Channel

Return to Glossary

See **Payment Channel**

Miner Node

Return to Glossary

A **Miner Node** on creates blocks in the blockchain which the nodes keep. Basically, the miner works on transactions by coming up with the best combination (hash) to store that information. In Bitcoin, Miners spend about 10 minutes working on a problem, but nodes keep that result forever after in the database and verify it with others. Miners don't need to know about prior blocks (except for the prior one) with very few exceptions.

So, a miner is completely different than a full node. It's not comparing the same like things. Full vs Light is comparing two like things - fruit (apple and orange). Miner vs FullNode is comparing two totally different things (apple and fence).

Source: Miner Node

Mission Assurance (MA)

Return to Glossary

Mission Assurance (MA) is the ability of operators to achieve their mission, continue critical processes, and protect people and assets in the face of internal and external attack (both physical and cyber), unforeseen environmental or operational changes, and system malfunctions.

Source: MITRE Systems Engineering Guide

Multi-Signature (multisig)

Return to Glossary

Multi-Signature (multisig) isis a wallet configuration that requires at least two keys to authorize a transaction. Commonly used by cryptocurrency exchanges to ensure funds can’t be moved by a rogue employee, multisig also has applications for end-users. If you’re seeking to enhance the security of your noncustodial bitcoin wallet, multi-signature might be the answer.

Source: How to Use Multisig to Keep Your Coins Ultra-Safe

N

Return to Glossary

Create a Glossary entry starting with 'N' **Word or Expression** → Add page

- Network Traffic Analyzer
- Node
- Node Network

Network Traffic Analyzer

Return to Glossary

Network_traffic_analyzer (NTA) intercepts, records and analyzes network traffic communications. The communications are scanned for patterns that indicate security threats and provides methods to respond to security threats.

Source: Network Traffic Analyzer

Node

Return to Glossary

A **Node** is an individual computer that participates as an equal within the Node Network. A node is sometimes referred to as a **peer**. Nodes receive input, perform one or more operations on those inputs; and returns an output.[144)]

Source: Node

144)
Not all DIDOs use “mining” as originally defined in Bitcoin as Proof of Work (PoW) and they may not need a full set blockchain transaction to verify transactions.

Node Network

Return to Glossary

A **Node Network** is a collection of computers (i.e., interconnected) communicating over a network of equally privileged computers (i.e., there are no central authoritative computers). The **Node network** is sometimes referred to as a **peer-to-peer (P2P) network**. [145) 146) 147) 148)]

145)
"What is a Bitcoin node?", Nate Eldredge, 14, December 2013, https://bitcoin.stackexchange.com/questions/18736/what-is-a-bitcoin-node
146)
"What are Ethereum Nodes And Sharding?", Ameer Rosic, 2017, https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/
147)
"IOTA Full Node—That's Why a Full Node Is Important for IOTA!", Marko Vidrih, February 2019, https://medium.com/altcoin-magazine/iota-full-node-thats-why-a-full-node-is-important-for-iota-1c46280d7712
148)
"Introduction to IPFS: Run Nodes on Your Network, with HTTP Gateways", Ross Bulat, 3 December 2018, https://medium.com/@rossbulat/introduction-to-ipfs-set-up-nodes-on-your-network-with-http-gateways-10e21ea689a4

O

Return to Glossary

Create a Glossary entry starting with 'O' **Word or Expression** →	Add page

- Open Source Software
- Operating System (OS)
- Operational transformation (OT)
- Oracle

Open Source Software (OSS)

Return to Glossary

Open source software (OSS) doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria: free distribution; source code; derived works; integrity of the author’s source code; no discrimination against persons or groups; no discrimination against fields of endeavor; distribution license; license must not be specific to a product; license must not restrict other software; and license must be technology-neutral.

Source: The Open Source Definition, 22 March 2007

Operating System (OS)

Return to Glossary

An **operating system (OS)**, in its most general sense, is software that allows a user to run other applications on a computing device. While it is possible for a software application to interface directly with hardware, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details.

Source: Operating System (OS)

Operational transformation (OT)

Return to Glossary

Operational transformation (OT) is a technology for supporting a range of collaboration functionalities in advanced collaborative software systems. OT was originally invented for consistency maintenance and concurrency control in collaborative editing of plain text documents. Its capabilities have been extended and its applications expanded to include group undo, locking, conflict resolution, operation notification and compression, group-awareness, HTML/XML and tree-structured document editing, collaborative office productivity tools, application-sharing, and collaborative computer-aided media design tools.[1] In 2009 OT was adopted as a core technique behind the collaboration features in Apache Wave and Google Docs.

Source: Operational transformation (OT)

Oracle

Return to Glossary

An **Oracle**, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts.

Source: Blockchain Oracles

P

Return to Glossary

Create a Glossary entry starting with 'P' **Word or Expression →** Add page

- Parliamentary Authority
- Payment Channel
- Pedigree
- Peer to Peer (P2P)
- Performance Efficiency Measure
- Permissioned Networks
- Permissionless Networks
- Permissive Open Source Software
- Platform-as-a-Service (PaaS)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)
- Policy
- Private Network
- Principle
- Principles
- Proof of Authority (PoA)
- Procedure
- Proof of Stake (PoS)
- Proof of Work (PoW)
- Provenance
- Public Network

Parliamentary Authority

Return to Glossary

Parliamentary Authority is the rulebook used to conduct business within a group. Robert's Rules of Order is not the only one, but it is a common option used to help facilitate the smooth functioning of an assembly and provide a firm basis for resolving questions of procedure.

Source: Parliamentary Authority

Payment Channel

Return to Glossary

Payment Channel is a medium between two or more parties that allows them to transact bitcoins amongst them a number of times without sending all these transactions to the base layer, i.e., Bitcoin's blockchain.

A payment channel essentially is a 2-of-2 Multi-Signature (multisig) wallet where parties interested in transacting with each other commit funds and this is called funding transaction that happens on-chain.

Once this on-chain transaction is complete, many transactions can happen between the two parties involved in the payment channel. But the transactions can only happen when both the parties sign thus updating the state of the channel (or you can say an offline shared ledger between them)

Thus many signed but broadcasted transactions can be exchanged between the parties involved in the payment channel.

In simple terms, you can think of a payment channel as duct or pipe which is open from both the ends and there are two individuals on each side with 10 pearls with them.

So through this duct or pipe, these guys can send the pearls to each other number of times and when they don't want to transact they can simply close the duct or pipe.

Source: Payment Channel

Pedigree

Return to Glossary

Pedigree is meta-data about a record of the ancestry of data and may include metric estimates about the reliability and confidence in the data. In other words, pedigree is about the “who” and “when” of ownership, transfer and transformation of data.

Source: OMG

Peer to Peer (P2P)

Return to Glossary

Peer to Peer (P2P) is a network where the “peers” are computer systems which are connected to each other via the Internet. Files can be shared directly between systems on the network without the need of a central server. In other words, each computer on a P2P network becomes a file server as well as a client.

Source: Peer to Peer (P2P)

Performance Efficiency Measure

Return to Glossary

Performance Efficiency Measure assesses characteristics that affect an application's response behavior and use of resources under stated conditions (ISO/IEC 25010). Performance Efficiency affects customer satisfaction, workforce productivity, application scalability, response-time degradation, and inefficient use of processing or storage resources. The Performance Efficiency of an application lies in each individual component's performance, as well as in the effect of each component on the behavior of the chain of components comprising a transaction in which it participates.

Source: OASIS A Brief Introduction to XACML - 14 March 2003

Permissioned Networks

Return to Glossary

Permissioned blockchains allow a mixed bag between the Public Network and Private Network with lots of customization options. The available options include allowing anyone to join the permissioned network after suitable verification of their identity, and allocation of select and designated permissions to perform only certain activities on the network. [149)]

149)

"Public, Private, Permissioned Blockchains Compared", Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

Permissionless Networks

Return to Glossary

Permissionless Networks require no permission to use it. In other words, there is no barrier to entry to use it. Anyone can run a node, run mining software/hardware, access a wallet and write data onto and transact within the blockchain (as long as they follow the rules of the blockchain). There is no way to censor anyone, ever, on the permissionless bitcoin blockchain.[150)]

150)

"Permissioned vs. Permissionless blockchains: Who will win and will it matter?", Dustin D, 22 March 2018, Permissionless

Permissive Open Source Software

Return to Glossary

Permissive Open Source Software is simply a non-copyleft open source license — one that guarantees the freedoms to use, modify, and redistribute, but that permits proprietary derivative works. See the copyleft entry for more information.

Source: What is a "permissive" Open Source license?

Platform-as-a-Service (PaaS)

Return to Glossary

Platform-as-a-Service (PaaS) is a form of cloud computing where the hardware and software platform is provided by a third party.

Source: Platform-as-a-Service (PaaS)

Platform Independent Model (PIM)

Return to Glossary

Platform Independent Model (PIM): A model of a subsystem that contains no information specific to the platform or the technology that is used to realize it. The quality that the model is independent of the features of a platform of any particular type. Like most qualities, platform independence is a matter of degree. A view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type. [MDA Guide]. Platform Independent Viewpoint: Focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another

Source: Platform-Independent Model (PIM)

Platform Specific Model (PSM)

Return to Glossary

**Platform Specific Model (PSM)*: A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform. A view of a system from the platform specific viewpoint, combining the specifications in the PIM with the details that specify how that system uses a particular type of platform Source: Platform-Specific Model (PSM)

Policy

Return to Glossary

Policy is a precise statement which contains the set of principles acting as guidelines for achieving the goals of an organization.

Source: Policy

Principle

Return to Glossary

A **Principle** is an elementary assumption, concept, doctrine, maxim, or proposition generally held to be fundamental or true for a body of knowledge, conduct, procedure, or system of reasoning, and used as a basis for prediction and action. See also principles.

Source: Principles

Principles

Return to Glossary

Principles are fundamental norms, rules, or values that represent what is desirable and positive for a person, group, organization, or community, and help it in determining the rightfulness or wrongfulness of its actions. Principles are more basic than policy and objectives, and are meant to govern both. See also Principle.

Source: Principles

Private Network

Return to Glossary

A **Private Network** allows only selected entry of verified participants, like those for a private business, one can opt for a private network implementation. A participant can join such a private network only through an authentic and verified invitation, and a validation is necessary either by the network operator(s) or by a clearly defined set protocol implemented by the network.[151)]

151)
“Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

Procedure

Return to Glossary

Procedure is a fixed, step-by-step sequence of activities or course of action (with definite start and end points) that must be followed in the same order to correctly perform a task. Repetitive procedures are called routines.

Source: Procedure

Proof of Authority (PoA)

Return to Glossary

Proof of Authority (PoA) is a modified form of Proof of Stake (PoS) where instead of stake with the monetary value, a validator's identity performs the role of stake. In this context, identity means the correspondence between a validator's personal identification on the platform with officially issued documentation for the same person, i.e. certainty that a validator is exactly who that person represents to be.

Source: Proof of Authority (PoA)

Proof of Stake (PoS)

Return to Glossary

Proof of Stake (PoS) is a way of achieving consensus by distributing validation of a block of transactions based on the number of tokens held rather than by rewarding miners like Proof of Work (PoW). Consequently, PoS is a way of validating a block that requires far less energy that Proof of Work (PoW). An important key aspect of PoS is including a degree of chance to the selection process to avoid a scenario where the richest users are always selected to validate transactions and consistently reap the rewards or to bias validation in their favour.

Source: What is Proof of Stake? (PoS)

Proof of Work (PoW)

Return to Glossary

Proof of work (PoW) describes a system that requires a not-insignificant but feasible amount of effort in order to deter frivolous or malicious uses of computing power, such as sending spam emails or launching denial of service attacks. The concept was adapted to money by Hal Finney in 2004 through the idea of “reusable proof of work.” [152] Following its introduction in 2009, bitcoin became the first widely adopted application of Finney's idea (Finney was also the recipient of the first bitcoin transaction). Proof of work forms the basis of many other cryptocurrencies as well.

Source: Proof of Work

152)

https://nakamotoinstitute.org/finney/rpow/index.html

Provenance

Return to Glossary

Provenance is meta-data about a record of the transformation of data such as inputs, entities, systems, and processes that influence data of interest. In other words, provenance is about the “how” and “what” of transfer, and transformation of data.

Source: OMG

Public Network

Return to Glossary

A **Public Network** is completely open and anyone is free to join and participate in the core activities of the blockchain network. Anyone can join or leave, read, write and audit the ongoing activities on the public blockchain network, which helps a public blockchain maintain its self-governed nature. [153)]

153)

“Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

Q

Return to Glossary

Create a Glossary entry starting with 'Q' **Word or Expression** →		Add page

R

Return to Glossary

Create a Glossary entry starting with 'R' **Word or Expression** → Add page

- Reference Architecture (RA)
- Registered Agent
- Relational DataBase Management System (RDBMS)
- Reliability Measure
- Request For Comment (RFC)
- Request For Information (RFI)
- Request For Proposal (RFP)
- Representational state transfer (REST)
- RESTful API
- Rich Site Summary (RSS)
- Risk

Reference Architecture (RA)

Return to Glossary

A **Reference Architecture** (RA) is defined as a set of goals:

- Provide common language for the various stakeholders
- Provide consistency of implementation of technology to solve problems
- Support the validation and comparison of implementations
- Encourage adherence to common standards, specifications, and patterns

Source: G. Doyle and B. Wilezynski, “Reference Architecture Description,” U. S. DoD, Washington, DC, 2010., Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII) Reference Architecture Description

Registered Agent

Return to Glossary

Registered Agent is a responsible third-party who is located in the same state in which a business entity was established and who is designated to receive service of process notices, correspondence from the Secretary of State, and other official government notifications, usually tax forms and notice of lawsuits, on behalf of the corporation or LLC.

Source: Registered Agent

Relational DataBase Management System (RDBMS)

Return to Glossary

Relational DataBase Management System (RDBMS) is a database engine/system based on the relational model specified by Edgar F. Codd–the father of modern relational database design–in 1970.

Most modern commercial and open-source database applications are relational in nature. The most important relational database features include an ability to use tables for data storage while maintaining and enforcing certain data relationships.

Source: Relational DataBase Management System (RDBMS)

Reliability Measure

Return to Glossary

The **Reliability Measure** the risk of potential application failures and the stability of an application when confronted with unexpected conditions. According to ISO/IEC/IEEE 24765, Reliability is the degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time. The reason for checking and monitoring Reliability is to prevent or at least reduce application downtime, outages, data corruption, and errors that directly affect users.

Source: OASIS Security Services (SAML) TC - May 2012

Representational state transfer (REST)

Return to Glossary

Representational state transfer (REST) is a distributed system framework that uses Web protocols and technologies. The REST architecture involves client and server interactions built around the transfer of resources. The Web is the largest REST implementation. Systems that conform to REST principles are referred to as RESTful.

Source: Representational state transfer (REST)

Request For Comment (RFC)

Return to Glossary

OMG

Request For Comment (RFC) is an alternative to the Technology Adoption Process allowing an OMG member to request OMG adoption of an uncontentious specification without requiring a Request for Proposals to be issued.

Source: OMG Request For Comment (RFC)

IETF

A **Request for Comments (RFC)** is a formal document drafted by the Internet Engineering Task Force (IETF) that describes the specifications for a particular technology. When an RFC is ratified, it becomes a formal standards document. RFCs were first used during the creation of the ARPANET protocols that came to establish what became today's Internet. They continue to be issued on an ongoing basis as the technology underlying the Internet evolves.

Source: IETF Request For Comment (RFC)

Request For Information (RFI)

Return to Glossary

Request For Information (RFI) A general request to the computer industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's TFs. Information received in response to an RFI is typically used by a TF to formulate one or more RFPs.

Source: Request For Information (RFI)

Request For Proposal (RFP)

Return to Glossary

Request For Proposal (RFP) is the requirements document for a new OMG technology specification. Issuance of an RFP starts the OMG Technology Adoption Process. RFPs are written and recommended by a TF, certified by the AB, and issued by vote of the TF's Parent Body.

Source: Request For Proposals (RFP)

RESTful API

Return to Glossary

RESTful API is an API that conforms to the representational state transfer or REST model. RESTful APIs are sometimes easier for developers to use because they have a familiar syntax and set of protocols. As more functionality has been built into the internet, developers have talked a lot about the benefits of RESTful architecture.

Source: RESTful API

Rich Site Summary (RSS)

Return to Glossary

Rss a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator automatically checks the RSS feed for new content, allowing the list to be automatically passed from website to website or from website to user.

Note: Also known as Really Simple Syndication (RSS).

Source: Rich Site Summary (RSS)

Risk

Return to Glossary

Risk is a probability or threat of damage, injury, liability, loss, or any other negative occurrence that is caused by external or internal vulnerabilities, and that may be avoided through preemptive action.

Source: Risk

S

Return to Glossary

Create a Glossary entry starting with 'S' **Word or Expression** → Add page

- Safety Assurance (SfA)
- Salami Slicing
- Sarbanes-Oxley Act (SOX)
- Security Measure
- Semantic Web
- Simple Payment Verification (SPV)
- Soft Fork
- Smart Contracts
- Software as a Service (SaaS)
- Software Assurance (SwA)
- Snapshot
- Stakeholder
- Standards Developing Organization (SDO)
- Special Interest Group (SIG)
- Special Rules
- Standards Organization
- Standing Rules
- Statute
- Straight-through Processing (StP)
- System Assurance
- Systems and software Quality Requirements and Evaluation (SQuaRE)

Safety Assurance (SfA)

Return to Glossary

Safety Assurance (SfA) is providing confidence that acceptable risk for the safety of personnel, equipment, facilities, and the public during and from the performance of operations is being achieved.

Source: FAA/NASA

Salami Slicing

Return to Glossary

Salami Slicing refers to a series of many small actions, often performed by clandestine means, that as an accumulated whole produces a much larger action or result that would be difficult or unlawful to perform all at once. The term is typically used pejoratively. Although salami slicing is often used to carry out illegal activities, it is only a strategy for gaining an advantage over time by accumulating it in small increments, so it can be used in legal ways as well.

An example of salami slicing, also known as penny shaving, is the fraudulent practice of stealing money repeatedly in extremely small quantities, usually by taking advantage of rounding to the nearest cent (or other monetary unit) in financial transactions. It would be done by always rounding down, and putting the fractions of a cent into another account. The idea is to make the change small enough that any single transaction will go undetected

Source: Salami Slicing

Sarbanes-Oxley Act (SOX)

Return to Glossary

The **Sarbanes-Oxley Act** (SOX) was designed to improve the quality of financial reporting by public companies. It was written in response to the fraudulent reporting of Enron Corporation, Worldcom, and several other businesses, and was passed in 2002. Key provisions of the Act are as follows:

- The CEO and CFO must certify the accuracy of the financial statements (Section 302).
- It is illegal to improperly influence how an audit is conducted (Section 303).
- Material off-balance sheet items must be disclosed (Section 401).
- Management must establish internal controls and report on their scope and accuracy, while the company's auditors must certify the reliability of those controls (Section 404).
- Substantial fines are imposed on anyone falsifying, stealing, or destroying records (section 802).
- Provides for the protection of whistleblowers from retaliation (Section 806).
- Sets criminal penalties when corporate officers do not certify the accuracy of the financial statements (Section 906).

The provisions of the Act made it significantly more expensive for firms to be publicly-held. The result was a decline in the number of public companies, especially among the smaller firms that could no longer afford the regulatory costs associated with being publicly-held. In particular, the requirements of Section 404 were considered to have the largest impact on the cost increase.

The official name of the Sarbanes-Oxley Act is the Corporate Responsibility Act of 2002.

Source: Sarbanes-Oxley Act

Security Measure

Return to Glossary

The **Security Measure** assesses the degree to which an application protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization (ISO 25010). Security measures the risk of potential security breaches due to poor coding and architectural practices. Security problems have been studied extensively by the Software Assurance community and have been codified in the Common Weakness Enumeration (CWE).

Source: OASIS eXtensible Access Control Markup Language (XACML) TC - 22 January 2013

Semantic Web

Return to Glossary

The Semantic Web is a mesh of data associated in such a way they can easily be processed by machines instead of human operators. It can be conceived as an extended version of the existing World Wide Web (WWW), and it represents an effective means of data representation in the form of a globally linked database. By supporting the inclusion of semantic content in Web pages, the Semantic Web targets the conversion of the presently available Web of unstructured documents to a Web of information/data.

Source: Semantic Web

Simple Payment Verification (SPV)

Return to Glossary

Simple Payment Verification (SPV) is a technique described in Satoshi Nakamoto's paper allowing a lightweight client to verify that a transaction is included in the Bitcoin blockchain, without downloading the entire blockchain. The client just needs to download the headers rather than the heavier full blocks.

Source: Simple Payment Verification (SPV)

Smart Contracts

Return to Glossary

Smart Contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist across a distributed, decentralized blockchain network.

Source: Smart Contracts

Snapshot

Return to Glossary

Snapshot is done to prevent the Tangle (DAG) from expanding too much in size. Snapshotting saves all the balances, while removing the history and data of all the transactions to start fresh. These addresses with balances act like a new genesis address, but no previous history or data will be attached.

Source: Snapshot

Soft Fork

Return to Glossary

a soft fork (or sometimes softfork) is a change to the software protocol where only previously valid blocks/transactions are made invalid. Since old nodes will recognize the new blocks as valid, a softfork is backward-compatible. This kind of fork requires only a majority of the miners upgrading to enforce the new rules, as opposed to a hard fork which requires all nodes to upgrade and agree on the new version.

Source: Soft Fork

Software as a Service (SaaS)

Return to Glossary

Software as a Service (SaaS) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. SaaS is one of three main categories of cloud computing, alongside Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS).

Source: Software as a service (SaaS)

Software Assurance (SwA)

Return to Glossary

Software Assurance (SwA) is (1) The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle (DoD); (2) The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures (NASA)

Source: Committee on National Security Systems, CNS 4009 Glossary

Special Interest Group (SIG)

Return to Glossary

Special Interest Group (SIG) is one of the three types of OMG Subgroups, a SIG may be chartered by any one of the three Plenary Bodies. Unlike Task Forces (TFs), SIGs may not recommend issuance of RFPs or adoption of technology to their Parent Body, nor may they issue RFIs although they may issue Surveys which serve the same function as an RFI but carry a different name. Typically, SIGs function as discussion groups and/or technology incubators, i.e, primary authors of RFPs or RFIs. Occasionally a SIG will be disbanded and its Parent Body will charter a TF with a similar name and function, typically after a period of several months to a year or more during which the SIG has demonstrated that its membership could sustain the effort necessary to recommend and maintain adopted technology.

Source: Special Interest Group (SIG)

Special Rules

Return to Glossary

Special Rules are rules that supplement or modify the group's chosen parliamentary authority.

Source: Special Rules

Stakeholder

Return to Glossary

Stakeholder is is a person, group or organization that has interest or concern in a *[target]* organization. Stakeholders can affect or be affected by the [target] organization's actions, objectives and policies.
Note: *[target]* added for clarification purposes

Source: Stakeholder

Standards Developing Organization (SDO)

Return to Glossary

Standards Developing Organization (SDO), Standards Organization, Standards Body, or Standards Setting Organization (SSO) is an organization whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards[1] that are intended to address the needs of a group of affected adopters.

Most standards are voluntary in the sense that they are offered for adoption by people or industry without being mandated in law. Some standards become mandatory when they are adopted by regulators as legal requirements in particular domains.

Source: Standards Developing Organization (SDO)

Standards Organization

Return to Glossary

Standards Organizations are organizations whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards that are intended to address the needs of a group of affected adopters. [154)]

Source: Standards Organization

154)

Intercloud: Solving Interoperability and Communication in a Cloud of Clouds, Appendix A, by Ken Owens, Monique Morrow, Venkata Josyula, Jazib Frahim, Publisher: Cisco Press Release Date: June 2016 ISBN: 9780134189239,https://learning.oreilly.com/library/view/intercloud-solving-interoperability/9780134189239/app01.html

Standing Rules

Return to Glossary

Standing Rules are regulations or rules that deal with the procedures and operations of a business or guidance of an institution or administration of a society and are adopted from time to time similarly as any other act of the deliberative assembly. Generally, the standing rules can be amended only by a majority vote.

Source: Standing Rules

Statute

Return to Glossary

Statute is a formally drafted and written law adopted by both chambers or houses of a legislature. Statutes are enacted usually by voting following an open discussion, and signed thereafter by the head of State and included in the country's statute book.

Source: Statute

Straight-through Processing (StP)

Return to Glossary

Straight-through Processing is an automated electronic payment process that is used by corporations and banks. STP allows for the entire payment process, from initiation to final settlement, to be free of human intervention. Straight-through processing can help local businesses, as well as large corporations, pay and receive money faster than the traditional process.

Source: Straight-through Processing (StP)

System Assurance (SysA)

Return to Glossary

System Assurance (SysA) is the planned and systematic set of engineering activities necessary to assure that products conform with all applicable system requirements for safety, security, reliability, availability, maintainability, standards, procedures, and regulations, to provide the user with acceptable confidence that the system behaves as intended in the expected operational context.

Source: OMG SysA Task Force

Systems and software Quality Requirements and Evaluation (SQuaRE)

Return to Glossary

Systems and software Quality Requirements and Evaluation (SQuaRE) is an extension (ISO/IEC 25050 to ISO/IEC 25099) and is designated to contain system or software product quality International Standards and/or Technical Reports that address specific application domains or that can be used to complement one or more SQuaRE International Standards.

Source: ISO/IEC 25010:2011

T

Return to Glossary

Create a Glossary entry starting with 'T' **Word or Expression** → Add page

- Tangle
- Taxonomy
- Technical Standard
- Tokens
- Transmission Control Protocol (TCP)

Tangle

Return to Glossary

Tangle is the moniker used to describe IOTA's Directed Acyclic Graph (DAG) based transaction settlement and data integrity layer focused on the Internet of Things (IOT). The Tangle is essentially a string of individual transactions that are interlinked to each other and stored through a decentralized network of node participants.

Importantly, the Tangle does not have miners as users of the network function as the miners themselves through performing small computational Proof of Work (PoW) for each transaction by verifying previous transactions submitted to the network. Focused on allowing the network to scale for a global micropayment network of interconnected IoT devices, The Tangle is designed to offer a solution to the heterogeneous nature of current blockchain systems.

Source: What is The Tangle? Complete Guide to IOTA's Directed Acyclic Graph (DAG)

Taxonomy

Return to Glossary

Taxonomy formalizes the hierarchical relationships among concepts and specifies the term to be used to refer to each; it prescribes structure and terminology.

- A taxonomy is a form of classification scheme
- Taxonomies are semantic
- A taxonomy is a kind of knowledge map

Sources:

- Patrick Lambe Defining Taxonomy and
- Taxonomy

Technical Standard

Return to Glossary

A **Technical Standard** is an established norm or requirement in regard to technical systems captured as formal document establishing uniform engineering or technical criteria, methods, processes, and practices. **Technical Standards** are defined by standards organizations.

Source: Technical Standard

Tokens

Return to Glossary

Tokens are digital assets, issued by the project, which can be used as a method of payment inside project's ecosystem, performing similar functions with Coins, but **the main difference is that it also gives the holder a right to participate in the network.** It may perform the functions of digital asset, represent a company's share, give access to the project's functional and many more—with the launching of new projects unknown facets of tokens' functional are discovered. Ticket to a concert, for example, is a "real-life token"—you may use it at a certain time, at a certain place. You can't go to the restaurant and pay your bill with concert ticket—ticket has its value only at concert hall. Digital tokens are the same—they have certain use case only inside certain project.

Tokens represent an asset or utility, so security and utility tokens are distinguished. Security tokens are designed to be the company's share (token of notorious project DAO, that was hacked right after launching, was recognized as security token), while utility tokens have certain use case inside the project (BON Token).

Creating a token is easier than creating a coin, as you don't have to create a new code or modify already existing one—you just use a standard template from platforms like Ethereum, that are blockchain-based and allow anyone to create tokens in just few steps. Using a template for creating tokens provides smooth interoperability, so users can store different types of tokens in one wallet. Ethereum was the first to simplify the process of creating a token, being not the last reason why tokens flooded the market.

Compare to Coins

Source: Tokens

Transmission Control Protocol (TCP)

Return to Glossary

Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other.

Source: Transmission Control Protocol (TCP)

U

Return to Glossary

Create a Glossary entry starting with 'U' **Word or Expression** →	Add page

- Unified Modeling Language (UML)
- UNIX

Unified Modeling Language (UML)

Return to Glossary

Unified Modeling Language (UML) is the OMG standard language for Analysis and Design of applications, specifying the structure and behavior of systems. UML is defined as an underlying abstract syntax and an overlying graphical concrete representation.

Source: Unified Modeling Language (UML)

UNIX

Return to Glossary

The UNIX brand has traditionally been applied to the family of multitasking, multiuser computer operating systems that derive from the original AT&T UNIX operating system, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others [155].

Today, the definition of UNIX ®[156] takes the form of the worldwide Single UNIX Specification integrating X/Open Company's XPG4, IEEE's POSIX Standards and ISO C. Through continual evolution, the Single UNIX Specification is the defacto and dejure standard definition for the UNIX system application programming interfaces. As the owner of the UNIX trademark, The Open Group has separated the UNIX trademark from any actual code stream itself, thus allowing multiple implementations. Since the introduction of the Single UNIX Specification, there has been a single, open, consensus specification that defines the requirements for a conformant UNIX system.

There is also a mark, or brand, that is used to identify those products that have been certified as conforming to the Single UNIX Specification, initially UNIX 93, followed subsequently by UNIX 95, UNIX 98, UNIX 03 and now UNIX V7.

155)
"The invention of Unix", Nokia Bell Labs, https://www.bell-labs.com/var/articles/invention-unix/
156)
"What is Unix", The Open Grup, http://www.unix.org/what_is_unix.html

V

Return to Glossary

Create a Glossary entry starting with 'V' **Word or Expression** →		Add page

- Virtual Machine (VM)

Virtual Machine (VM)

Return to Glossary

Virtual Machine (VM) is a software program or operating system that not only exhibits the behavior of a separate computer, but is also capable of performing tasks such as running applications and programs like a separate computer.

In other words, a VM is a software application that performs most functions of a physical computer, actually behaving as a separate computer system.

A virtual machine, usually known as a guest, is created within another computing environment referred as a “host.” Multiple virtual machines can exist within a single host at one time.

Source: Virtual Machine (VM)

W

Return to Glossary

Create a Glossary entry starting with 'W' **Word or Expression** →		Add page

- Weight of Network

Weight of Network

Return to Glossary

Weight of Network is a figure given to all the coins that are actively Staking on the entire XLR Network. The Coins in your Wallet that are available for Staking also has a Weight. If the Weight of the Network is 8000 and the Weight of your Coins is 2000 then you have a good chance of Minting some new coins. However if the Weight of your Coins is just 1 then the chance of you Minting some new coins are remote. Some Networks combine other factors to the calculation of weight such as age of the coins (i.e., older coins are heavier).

Source: Masternode vs Staking

X

Return to Glossary

Create a Glossary entry starting with 'X' **Word or Expression** →		Add page

Y

Return to Glossary

Create a Glossary entry starting with 'Y' **Word or Expression** → Add page

Z

Return to Glossary

Create a Glossary entry starting with 'Z' **Word or Expression** →		Add page

Appendix B: Standards Organizations

return to the Public Area **OR** return to Reference Architecture (RA)

The DIDO RA recognizes two categories of standard bodies:

- Technical Standards Bodies
- de facto Standard Bodies

Technical Standards Bodies

return to Standards Area

Technical standards are developed by Standards Organizations [157]. Standards organizations are organizations whose primary activities are to develop, coordinate, promulgate, revise, amend, reissue, interpret, or otherwise produce technical standards intended to address the needs of a group of affected adopters.[158]

The DIDO RA provides data sheets for technical standards developed by the following organizations:

- Apache Software Foundation (ASF)
- ECMA International
- Institute of Electrical and Electronics Engineers (IEEE)
- Internet Engineering Task Force (IETF)
- International Organization for Standardization (ISO)
- International Telecommunications Union (ITU)
- National Institute of Standards and Technology (NIST)
- Organization for the Advancement of Structured Information Standards (OASIS)
- Object Management Group (OMG)
- Open Source Initiative (OSI)
- World Wide Web Consortium (W3C)

157)
also known as standards body, Standards Developing Organization (SDO), or Standards Setting Organization (SSO)

158)
Intercloud: Solving Interoperability and Communication in a Cloud of Clouds, Appendix A, by Ken Owens, Monique Morrow, Venkata Josyula, Jazib Frahim, Publisher: Cisco Press Release Date: June 2016 ISBN: 9780134189239,https://learning.oreilly.com/library/view/intercloud-solving-interoperability/9780134189239/app01.html

Apache Software Foundation (ASF)

return to the Technical Standards area

Source: About the Apache Software Foundation (ASF)

> *The mission of the Apache Software Foundation (ASF) is to provide software for the public good. We do this by providing services and support for many like-minded software project communities consisting of individuals who choose to participate in ASF activities.*
> *committers. Through the ASF's meritocratic process known as "The Apache Way," more than 730 individual Members and 7,000 Committers successfully collaborate to develop freely available enterprise-grade software, benefiting millions of users worldwide: thousands of software solutions are distributed under the Apache License; and the community actively participates in ASF mailing lists, mentoring initiatives, and ApacheCon, the Foundation's official user conference, trainings, and expo.*

Note Although the ASF is a Corporate Project and thus, strictly speaking a de facto standards organization, the Apache License is part of the larger Open Source Initiative (OSI) and thus, considered a Technical vice *de facto* Standard.

Technical Standards

- Apache License, Version 2.0 (Apache-2.0)

Apache License, Version 2.0 (Apache-2.0)

return to the Open Source Initiative area or return to Apache Software Foundation

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://www.apache.org/licenses/LICENSE-2.0.html

The 2.0 version of the Apache License, approved by the ASF in 2004, helps us achieve our goal of providing reliable and long-lived software products through collaborative open source software development.

All packages produced by the ASF are implicitly licensed under the Apache License, Version 2.0, unless otherwise explicitly stated.

License Metadata

License Attribute	Value
Full Name	Apache License, Version 2.0
SPDX Short Name	Apache-2.0
Author	Apache Software Foundation (ASF)
Latest Version	2.0
Publication Date	January 2004
Author URI	http://www.apache.org/licenses/
OSI URI	https://opensource.org/licenses/Apache-2.0
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

- Wikipedia - Comparison of free and open-source software licenses
- Permissive and Copyleft Are Not Antonyms

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" *shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.*

"Licensor" *shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.*

"Legal Entity" *shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition,* ***"control"*** *means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.*

"You" *(or* ***"Your"****) shall mean an individual or Legal Entity exercising permissions granted by this License.*

"Source" *form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.*

"Object" *form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.*

"Work" *shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).*

"Derivative Works" *shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.*

"Contribution" *shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition,* ***"submitted"*** *means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as* ***"Not a Contribution"****..*

"Contributor" *shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.*

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

> *1. You must give any other recipients of the Work or Derivative Works a copy of this License; and*
> *2. You must cause any modified files to carry prominent notices stating that You changed the files; and*
> *3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and*
> *4. If the Work includes a **"NOTICE"** text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.*

You may add Your own copyright statement to Your modifications and may provide additional or

different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file

7. Disclaimer of Warranty.

*Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an **"AS IS"** BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.*

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional

liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

*To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets **"[]"** replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same **"printed page"** as the copyright notice for easier identification within third-party archives.*

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the **"License"**);
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an **"AS IS"** BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

ECMA International

return to the Technical Standards area

Create a New Page for ECMA **Number** →		Add page

Ecma International is an industry association founded in 1961, dedicated to the standardization of information and communication systems.

Ecma has been involved in the development of the concept of “fast tracking” in ISO/IEC JTC 1 and, since its introduction in 1986, it has been one of the main practitioners of it.

Ecma is driven by industry to meet the needs of industry, generating a healthy competitive landscape based on differentiation of products and services, rather than technology models, generating confidence among vendors and users of new technology. About ECMA

Technical Standards

- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics

ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)

return to the Ecma Standards

Table 3: Data sheet for ECMAScript® 2018 Language Specification

Title	EECMAScript® 2018 Language Specification
Acronym	ECMAScript
Version	9
Series	TR
Document Number	ECMA-262
Release Date	June 2018
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

ECMAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific objects, whose description and behaviour are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

ECMAScript was originally designed to be used as a scripting language, but has become widely used as a general purpose programming language. A scripting language is a programming language that is used to manipulate, customize, and automate the facilities of an existing system. In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control. In this way, the existing system is said to provide a host environment of objects and facilities, which completes the capabilities of the scripting language. A scripting language is intended for use by both professional and nonprofessional programmers.

ECMAScript was originally designed to be a Web scripting language, providing a mechanism to enliven Web pages in browsers and to perform server computation as part of a Web-based client-server architecture. ECMAScript is now used to provide core scripting capabilities for a variety of

host environments. Therefore the core language is specified in this document apart from any particular host environment. ECMAScript usage has moved beyond simple scripting and it is now used for the full spectrum of programming tasks in many different environments and scales. As the usage of ECMAScript has expanded, so has the features and facilities it provides. ECMAScript is now a fully featured general-purpose programming language.

Some of the facilities of ECMAScript are similar to those used in other programming languages; in particular C, Java™, Self, and Scheme.

ECMA: Standard ECMA-334 - C# Language Specification

return to the Ecma Standards

Table 4: Data sheet for C# Language Specification

Title	C# Language Specification
Acronym	C#
Version	5
Series	Stansards
Document Number	ECMA-334
Release Date	December 2017
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

This specification describes the form and establishes the interpretation of programs written in the C# programming language. It describes:

- *The representation of C# programs;*
- *The syntax and constraints of the C# language;*
- *The semantic rules for interpreting C# programs;*
- *The restrictions and limits imposed by a conforming implementation of C#.*

This specification does not describe:

- *The mechanism by which C# programs are transformed for use by a data-processing system;*
- *The mechanism by which C# applications are invoked for use by a data-processing system;*
- *The mechanism by which input data are transformed for use by a C# application;*
- *The mechanism by which output data are transformed after being produced by a C# application;*
- *The size or complexity of a program and its data that will exceed the capacity of any specific data-processing system or the capacity of a particular processor;*

All minimal requirements of a data-processing system that is capable of supporting a conforming implementation.

ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)

return to the Ecma Standards

Table 5: Data sheet for C# Language Specification

Title	Common Language Infrastructure (CLI)
Acronym	CLI
Version	6
Series	Stansards
Document Number	ECMA-335
Release Date	June 2012
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

This Standard defines the Common Language Infrastructure (CLI) in which applications written in multiple high-level languages can be executed in different system environments without the need to rewrite those applications to take into consideration the unique characteristics of those environments. This Standard consists of the following parts:

- *Partition I: Concepts and Architecture – Describes the overall architecture of the CLI, and provides the normative description of the Common Type System (CTS), the Virtual Execution System (VES), and the Common Language Specification (CLS). It also provides an informative description of the metadata.*
- *Partition II: Metadata Definition and Semantics – Provides the normative description of the metadata: its physical layout (as a file format), its logical contents (as a set of tables and their relationships), and its semantics (as seen from a hypothetical assembler, ilasm).*
- *Partition III: CIL Instruction Set – Describes the Common Intermediate Language (CIL) instruction set.*
- *Partition IV: Profiles and Libraries – Provides an overview of the CLI Libraries, and a specification of their factoring into Profiles and Libraries. A companion file, CLILibrary.xml, considered to be part of this Partition, but distributed in XML format, provides details of each class, value type, and interface in the CLI Libraries.*
- *Partition V: Debug Interchange Format – Describes a standard way to interchange debugging information between CLI producers and consumers.*
- *Partition VI: Annexes – Contains some sample programs written in CIL Assembly Language (ILAsm), information about a particular implementation of an assembler, a machine-readable description of*

the CIL instruction set which can be used to derive parts of the grammar used by this assembler as well as other tools that manipulate CIL, a set of guidelines used in the design of the libraries of Partition IV, and portability considerations.

ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File

return to the Ecma Standards

Table 6: Data sheet for Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File

Title	Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
Acronym	
Version	6
Series	TR
Document Number	TR/84
Release Date	June 2012
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-TR/ECMA%20TR-084.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

This Technical Report is intended as an aid for understanding the libraries specified in Standard ECMA-335, Partition IV: Profiles and Libraries. That Partition includes a machine-readable specification, in XML, of the types that comprise the standard libraries. This Technical Report, in companion files, provides the following items which help to form a traceable chain from the normative XML specification to a portable, printable representation of its contents:

- ***Tool Source Code:*** *A program written in the C# programming language, XML Style-sheet Language (XSL), and using the facilities of the Microsoft .NET Framework™ and Microsoft Office™ to convert the XML into files viewable using Microsoft Word™. This program, initially provided by Intel Corporation and updated by the CLI editor for this edition, can be modified to produce other views of the XML.*
- ***Microsoft Word™ Files:*** *These are the files produced by running the tool mentioned above on the XML from Partition IV. The Ecma task group TC49/TG3 used similar files (produced using earlier versions of this tool run against earlier versions of the XML) as the primary means of reviewing the XML.*
- ***PDF™ Files:*** *These files are produced from the Microsoft Word™ files using the Adobe Acrobat™ program. They are viewable on a wide range of computer systems and printable on a range of computer output devices. In most cases, they will appear visually identical regardless of the means*

used to render them.

ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics

return to the Ecma Standards

Table 7: Data sheet for Common Language Infrastructure (CLI) - Common Generics

Title	Common Language Infrastructure (CLI) - Common Generics
Acronym	
Version	2
Series	TR
Document Number	TR/89
Release Date	June 2006
About Specification	
Download	https://www.ecma-international.org/publications/files/ECMA-TR/ECMA%20TR-089.pdf

Note: The following is an excerpt from the official ECMA description. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The CLI standard libraries (ISO/IEC 23271) provide a collection of common types that can be used by multiple languages. With the addition of generics to the CLI, the standard libraries have been extended to include a number of common generic types, in particular, collections. However, at present, these libraries do not include many simple generic types found in a number of different languages. Any language which uses these common types must implement them rather than deferring to the CLI library, thereby reducing language interoperability. This proposal addresses this issue by providing a number of these common types.

Generic tuples (product types) are standard in a number of languages: C++ (`template Pair`), Ada, Haskell, and Standard ML (SML). However, languages differ in the number of predefined tuple sizes supported by their standard libraries; e.g., C++ provides just one (Pair) while Haskell provides eight (sizes 2 to 9) and SML allows any size of tuple. This proposal provides nine (sizes 2 to 10).

Generic programming encourages "higher order" programming where generic functions (methods) take function (delegate) type arguments that have generic types. Examples include Ada's with and generic constraints, and function arguments in Haskell and SML. In the CLI, function values are provided in the form of delegates, so this proposal defines standard generic delegate types for functions (which return a value) and procedures (which do not).

Another two types that occur in a number of languages are an optional type, which either contains a value of some other type or an indication that such a value is not present; and an either type, which holds a value of one of two possible types and an indication of which one is present. This

proposal provides both of these. (Note: The optional type is similar to, but different from, the type `System.Nullable`.)

Finally, in existing generic languages, a need has been found for a filler type to be used when a particular generic parameter is not required for a particular use of the generic type. A standard one-value type is often provided for this purpose, often called `Unit` or `Void` . This proposal includes such a type.

Institute of Electrical and Electronics Engineers (IEEE)

return to the Technical Standards area

Create a New Page for IEEE **Number**→	Add page

Source: About IEEE

IEEE Standards Association (IEEE-SA) is a leading consensus building organization that nurtures, develops and advances global technologies, through IEEE. We bring together a broad range of individuals and organizations from a wide range of technical and geographic points of origin to facilitate standards development and standards related collaboration. With collaborative thought leaders in more than 160 countries, we promote innovation, enable the creation and expansion of international markets and help protect health and public safety. Collectively, our work drives the functionality, capabilities and interoperability of a wide range of products and services that transform the way people live, work, and communicate.

IEEE-SA is governed by the Board of Governors (BOG) who are elected by IEEE-SA Members. The Board of Governors oversees number of committees that are dedicated to manage key operational aspects of the IEEE-SA. The IEEE-SA Standards Board reports directly to the BOG, and oversees the IEEE standards development process. Standards Board members are elected by IEEE-SA members as a privilege of membership, and all Board Members and Committee members must be IEEE-SA members in good standing.

The IEEE-SA standards development process is open to members and non-members, alike. However, IEEE-SA membership enables standards development participants to engage in the standards development process at a deeper and more meaningful level, by providing additional balloting and participation opportunities. IEEE-SA members are the driving force behind the development of standards, providing technical expertise and innovation, driving global participation, and pursuing the ongoing advancement and promotion of new concepts.

IEEE-SA also engages and collaborates with global, regional, and national organizations from around the world in order to ensure the effectiveness and high visibility of IEEE standards within IEEE and the global community.

Technical Standards

- IEEE 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications

IEEE 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications

return to the IEEE Standards

Table 8: Data sheet for IEEE Standard for Information Technology–Portable Operating System Interface

Title	IEEE Standard for Information Technology–Portable Operating System Interface
Acronym	POSIX
Version	2017
Document Number	IEEE 1003.1-2017
Approved Date	2017-12-06
Published Date	2018-01-31
Reference	https://standards.ieee.org/standard/1003_1-2017.html
	http://pubs.opengroup.org/onlinepubs/9699919799/toc.htm

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Standard Details

POSIX.1-2017 defines a standard operating system interface and environment, including a command interpreter (or "shell"), and common utility programs to support applications portability at the source code level. It is intended to be used by both application developers and system implementors.

POSIX.1-2017 comprises four major components (each in an associated volume):

1. *General terms, concepts, and interfaces common to all volumes of POSIX.1-2017, including utility conventions and C-language header definitions, are included in the Base Definitions volume of POSIX.1-2017.*
2. *Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume of POSIX.1-2017.*
3. *Definitions for a standard source code-level interface to command interpretation services (a "shell") and common utility programs for application programs are included in the Shell and Utilities volume of POSIX.1-2017.*
4. *Extended rationale that did not fit well into the rest of the document structure, containing historical information concerning the contents of POSIX.1-2017 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume of POSIX.1-2017.*

The following areas are outside of the scope of POSIX.1-2017:

- *Graphics interfaces*
- *Database management system interfaces*
- *Record I/O considerations*
- *Object or binary code portability*
- *System configuration and resource availability*

POSIX.1-2017 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

The facilities provided in POSIX.1-2017 are drawn from the following base documents:

- *IEEE Std 1003.1, 2004 Edition (POSIX-1) (incorporating IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002, and IEEE Std 1003.1-2001/Cor 2-2004)*
- *The Open Group Technical Standard, 2006, Extended API Set Part 1*
- *The Open Group Technical Standard, 2006, Extended API Set Part 2*
- *The Open Group Technical Standard, 2006, Extended API Set Part 3*
- *The Open Group Technical Standard, 2006, Extended API Set Part 4*
- *ISO/IEC 9899:1999, Programming Languages - C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3*

Emphasis has been placed on standardizing existing practice for existing users, with changes and additions limited to correcting deficiencies in the following areas:

- *Issues raised by Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and ISO/IEC defect reports against ISO/IEC 9945*
- *Issues raised in corrigenda for The Open Group Technical Standards and working group resolutions from The Open Group*
- *Issues arising from ISO TR 24715:2006, Conflicts between POSIX and the LSB*
- *Changes to make the text self-consistent with the additional material merged*
- *Features, marked Legacy or obsolescent in the base documents, have been considered for removal in this version*
- *A review and reorganization of the options within the standard*
- *Alignment with the ISO/IEC 9899:1999 standard, including ISO/IEC 9899:1999/Cor.2:2004(E)*

Internet Engineering Task Force (IETF)

return to the Technical Standards area

Create a New Page for IETF RFC **Number** →		Add page

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. [159)]

Technical Standards

- RFC0147 - The Definition of a Socket
- RFC0768 - User Datagram Protocol (UDP)
- RFC0791 - Internet Protocol (IPv4)
- RFC0793 - Transmission Control Protocol
- RFC1034 - Domain Names - Concepts and Facilities
- RFC1035 - Domain Names - Implementation and Specification
- RFC1112 - Host Extensions for IP Multicasting
- RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC)
- RFC2026 - The Internet Standards Process
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- RFC2246 - The TLS Protocol
- RFC2315 - Cryptographic Message Syntax
- RFC2426 - vCard MIME Directory Profile
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC3339 - Date and Time on the Internet: Timestamps
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC3596 - DNS Extension to support IP Version 6
- RFC4122 - A Universally Unique IDentifier (UUID) URN Namespace
- RFC5011 - Automated Updates of DNS Security (DNSSEC) Trust Anchors
- RFC5424 - The Syslog Protocol (SYSLOG)
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC6376 - DomainKeys Identified Mail (DKIM) Signatures
- RFC6455 - The WebSocket Protocol
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC6891 - Extension Mechanisms for DNS (EDNS(0))
- RFC6979 - Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)
- RFC7061 - eXtensible Access Control Markup Language (XACML) XML Media Type

- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format

159)

The Internet Engineering Task Force (IETF) is the premier Internet standards body, developing open standards through open processes. https://www.ietf.org/about/

RFC0147 - The Definition of a Socket

return to the IETF Standards

Table 9: Data sheet for RFC0147 - The Definition of a Socket

Title	The Definition of a Socket
Acronym	
Version	1971
Document Number	RFC0147
Release Date	May 1971
Reference	https://tools.ietf.org/html/rfc0147

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

A socket is defined to be the unique identification to or from which information is transmitted in the network. The socket is specified as a 32 bit number with even sockets identifying receiving sockets and odd sockets identifying sending sockets. A socket is also identified by the host in which the sending or receiving processor is located.

RFC0768 - User Datagram Protocol (UDP)

return to the IETF Standards

Table 10: Data sheet for RFC0768 - User Datagram Protocol (UDP)

Title	User Datagram Protocol
Acronym	UDP
Version	1980
Document Number	RFC7068
Release Date	28 August 1980
Reference	https://tools.ietf.org/html/rfc768

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [160] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP). [161]

160)
Postel, J., “Internet Protocol,” RFC 760, USC/Information Sciences Institute, January 1980.
161)
Postel, J.,“Transmission Control Protocol,” RFC 761, USC/Information Sciences Institute, January 1980.

RFC0791 - Internet Protocol (IPv4)

return to the IETF Standards

Table 11: Data sheet for RFC0791 - Internet Protocol (IP)

Title	Transmission Control Protocol
Acronym	IP or IPv4
Version	1981
Document Number	RFC0791
Release Date	September 1981
Reference	https://tools.ietf.org/html/rfc791

> **Note**: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

Motivation

> *The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. Such a system has been called a "catenet" [162]. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.*

[162]
Cerf, V., "The Catenet Model for Internetworking," Information Processing Techniques Office, Defense Advanced Research Projects Agency, IEN 48, July 1978.

RFC0793 - Transmission Control Protocol

return to the IETF Standards

Table 12: Data sheet for RFC0793 - Transmission Control Protocol (TCP)

Title	Transmission Control Protocol
Acronym	TCP
Version	1981
Document Number	RFC0793
Release Date	September 1981
Reference	https://tools.ietf.org/html/rfc793

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

This document describes the functions to be performed by the Transmission Control Protocol, the program that implements it, and its interface to programs or users that require its services.

RFC1034 - Domain Names - Concepts and Facilities

return to the IETF Standards

Table 13: Data sheet for RFC1034 Domain Names - Concepts and Facilities (DNS)

Title	Domain Names - Concepts and Facilities
Acronym	DNS
Version	1987
Document Number	RFC1034
Release Date	November 1987
Reference	https://tools.ietf.org/html/rfc1034

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This RFC introduces domain style names, their use for Internet mail and host address support, and the protocols and servers used to implement domain name facilities.

RFC1035 - Domain Names - Implementation and Specification

return to the IETF Standards

Table 14: Data sheet for RFC1935 Domain Names - Implementation and Specification

Title	Domain Names - Implementation and Specification
Acronym	
Version	1987
Document Number	RFC1035
Release Date	November 1987
Reference	https://tools.ietf.org/html/rfc1035

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The goal of domain names is to provide a mechanism for naming resources in such a way that the names are usable in different hosts, networks, protocol families, internets, and administrative organizations.

From the user's point of view, domain names are useful as arguments to a local agent, called a resolver, which retrieves information associated with the domain name. Thus a user might ask for the host address or mail information associated with a particular domain name. To enable the user to request a particular type of information, an appropriate query type is passed to the resolver with the domain name. To the user, the domain tree is a single information space; the resolver is responsible for hiding the distribution of data among name servers from the user.

From the resolver's point of view, the database that makes up the domain space is distributed among various name servers. Different parts of the domain space are stored in different name servers, although a particular data item will be stored redundantly in two or more name servers. The resolver starts with knowledge of at least one name server. When the resolver processes a user query it asks a known name server for the information; in return, the resolver either receives the desired information or a referral to another name server. Using these referrals, resolvers learn the identities and contents of other name servers. Resolvers are responsible for dealing with the distribution of the domain space and dealing with the effects of name server failure by consulting redundant databases in other servers.

Name servers manage two kinds of data. The first kind of data held in sets called zones; each zone is the complete database for a particular “pruned” subtree of the domain space. This data is called authoritative. A name server periodically checks to make sure that its zones are up to date, and if not, obtains a new copy of updated zones from master files stored locally or in another name server. The second kind of data is cached data which was acquired by a local resolver. This data

may be incomplete, but improves the performance of the retrieval process when non-local data is repeatedly accessed. Cached data is eventually discarded by a timeout mechanism.

This functional structure isolates the problems of user interface, failure recovery, and distribution in the resolvers and isolates the database update and refresh problems in the name servers.

RFC1112 - Host Extensions for IP Multicasting

return to the IETF Standards

Table 15: Data sheet for Host Extensions for IP Multicasting

Title	Host Extensions for IP Multicasting
Acronym	
Version	1989
Document Number	RFC1112
Release Date	August 1989
Reference	https://tools.ietf.org/html/rfc1112

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

IP multicasting is the transmission of an IP datagram to a "host group", a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams, i.e., the datagram is not guaranteed to arrive intact at all members of the destination group or in the same order relative to other datagrams.

The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. A host need not be a member of a group to send datagrams to it.

A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address, not the membership of the group, that is permanent; at any time a permanent group may have any number of members, even zero. Those IP multicast addresses that are not reserved for permanent groups are available for dynamic assignment to transient groups which exist only as long as they have members.

Internetwork forwarding of IP multicast datagrams is handled by "multicast routers" which may be co-resident with, or separate from, internet gateways. A host transmits an IP multicast datagram as a local network multicast which reaches all immediately-neighboring members of the destination host group. If the datagram has an IP time-to-live greater than 1, the multicast router(s) attached to the local network take responsibility for forwarding it towards all other networks that have members of the destination group. On those other member networks that are reachable within the IP time-to-live, an attached multicast router completes delivery by transmitting the datagram as a local multicast.
This memo specifies the extensions required of a host IP implementation to support IP multicasting,

where a “host” is any internet host or gateway other than those acting as multicast routers. The algorithms and protocols used within and between multicast routers are transparent to hosts and will be specified in separate documents. This memo also does not specify how local network multicasting is accomplished for all types of network, although it does specify the required service interface to an arbitrary local network and gives an Ethernet specification as an example. Specifications for other types of network will be the subject of future memos.

RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC)

return to the IETF Standards

Table 16: Data sheet for RFC1831 emote Procedure Call Protocol Specification Version 2 (RPC)

Title	emote Procedure Call Protocol Specification Version 2
Acronym	RPC
Version	2
Document Number	RFC1831
Release Date	August 1995
Reference	https://tools.ietf.org/html/rfc1831

> **Note**: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This document specifies version two of the message protocol used in ONC Remote Procedure Call (RPC). The message protocol is specified with the eXternal Data Representation (XDR) language [9]. This document assumes that the reader is familiar with XDR. It does not attempt to justify remote procedure calls systems or describe their use. The paper by Birrell and Nelson [1] is recommended as an excellent background for the remote procedure call concept.

RFC2026 - The Internet Standards Process

return to the IETF Standards

Table 17: Data sheet for RFC2026 The Internet Standards Process

Title	The Internet Standards Process
Acronym	
Version	Version 3
Document Number	RFC2026
Release Date	October 1996
Reference	https://tools.ietf.org/html/rfc2026

> **Note**: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This memo documents the process used by the Internet community for the standardization of protocols and procedures. It defines the stages in the standardization process, the requirements for moving a document between stages and the types of documents used during this process. It also addresses the intellectual property rights and copyright issues associated with the standards process.

RFC2104 - Keyed-Hashing for Message Authentication (HMAC)

return to the IETF Standards

Table 18: Data sheet for Keyed-Hashing for Message Authentication (HMAC)

Title	Keyed-Hashing for Message Authentication
Acronym	HMAC
Version	1997
Document Number	RFC2104
Release Date	February 1997
Reference	https://tools.ietf.org/html/rfc2104

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such integrity check based on a secret key are usually called "message authentication codes" (MAC). Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties. In this document we present such a MAC mechanism based on cryptographic hash functions. This mechanism, called HMAC, is based on work by the authors [163)] where the construction is presented and cryptographically analyzed. We refer to that work for the details on the rationale and security analysis of HMAC, and its comparison to other keyed-hash methods.
HMAC can be used in combination with any iterated cryptographic hash function. MD5 and SHA-1 are examples of such hash functions. HMAC also uses a secret key for calculation and verification of the message authentication values. The main goals behind this construction are

- *To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available.*
- *To preserve the original performance of the hash function without incurring a significant degradation.*
- *To use and handle keys in a simple way.*
- *To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function.*
- *To allow for easy replaceability of the underlying hash function in case that faster or more secure hash functions are found or required.*

This document specifies HMAC using a generic cryptographic hash function (denoted by H). Specific instantiations of HMAC need to define a particular hash function. Current candidates for such hash functions include SHA-1 [164], MD5 [165], RIPEMD-128/160 [166]. These different realizations of HMAC will be denoted by HMAC-SHA1, HMAC-MD5, HMAC-RIPEMD, etc.

***Note:** To the date of writing of this document MD5 and SHA-1 are the most widely used cryptographic hash functions. MD5 has been recently shown to be vulnerable to collision search attacks [167]. This attack and other currently known weaknesses of MD5 do not compromise the use of MD5 within HMAC as specified in this document (see Dobb); however, SHA-1 appears to be a cryptographically stronger function. To this date, MD5 can be considered for use in HMAC for applications where the superior performance of MD5 is critical. In any case, implementers and users need to be aware of possible cryptoanalytic developments regarding any of these cryptographic hash functions, and the eventual need to replace the underlying hash function. (See section 6 for more information on the security of HMAC.)*

163)

M. Bellare, R. Canetti, and H. Krawczyk, “Keyed Hash Functions and Message Authentication”, Proceedings of Crypto'96, LNCS 1109, pp. 1-15. (http://www.research.ibm.com/security/keyed-md5.html)

164)

NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.

165)

Rivest, R., “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.

166)

H. Dobbertin, A. Bosselaers, and B. Preneel, “RIPEMD-160: A strengthened version of RIPEMD”, Fast Software Encryption, LNCS Vol 1039, pp. 71-82. ftp://ftp.esat.kuleuven.ac.be/pub/COSIC/bosselae/ripemd/

167)

H. Dobbertin, “The Status of MD5 After a Recent Attack”, RSA Labs' CryptoBytes, Vol. 2 No. 2, Summer 1996. http://www.rsa.com/rsalabs/pubs/cryptobytes.html

RFC2246 - The TLS Protocol

return to the IETF Standards

Table 19: Data sheet for The TLS Protocol (TLS)

Title	TThe TLS Protocol
Acronym	TLS
Version	1.0
Document Number	RFC2246
Release Date	January 1999
Reference	https://tools.ietf.org/html/rfc2246

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP[TCP]), is the TLS Record Protocol. The TLS Record Protocol provides connection security that has two basic properties:

- *The connection is private. Symmetric cryptography is used for data encryption (e.g., DES DES , RC4 RC4, etc.) The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption.*
- *The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA, MD5, etc.) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.*

The TLS Record Protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. The TLS Handshake Protocol provides connection security that has three basic properties:

- *The peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA RSA, DSS DSS, etc.). This authentication can be made optional, but is generally required for at least one of the peers.*

- *The negotiation of a shared secret is secure: the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection.*
- *The negotiation is reliable: no attacker can modify the negotiation communication without being detected by the parties to the communication.*

One advantage of TLS is that it is application protocol independent. Higher level protocols can layer on top of the TLS Protocol transparently. The TLS standard, however, does not specify how protocols add security with TLS; the decisions on how to initiate TLS handshaking and how to interpret the authentication certificates exchanged are left up to the judgment of the designers and implementors of protocols which run on top of TLS.

RFC2315 - Cryptographic Message Syntax

return to the IETF Standards

Table 20: Cryptographic Message Syntax (PKCS)

Title	Cryptographic Message Syntax
Acronym	PKCS
Version	1981
Document Number	RFC2315
Release Date	March 1998
Reference	https://tools.ietf.org/html/rfc2315

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

This document describes a general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes. The syntax admits recursion, so that, for example, one envelope can be nested inside another, or one party can sign some previously enveloped digital data. It also allows arbitrary attributes, such as signing time, to be authenticated along with the content of a message, and provides for other attributes such as countersignatures to be associated with a signature. A degenerate case of the syntax provides a means for disseminating certificates and certificate-revocation lists.

RFC2426 - vCard MIME Directory Profile

return to the IETF Standards

Table 21: Data sheet for RFC2426 vCard MIME Directory Profile (MIME-DIR)

Title	vCard MIME Directory Profile
Acronym	MIME-DIR
Version	1.0
Document Number	RFC2426
Release Date	September 1998
Reference	https://tools.ietf.org/html/rfc2426

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The [MIME-DIR] document defines a MIME Content-Type for holding different kinds of directory information. The directory information can be based on any of a number of directory schemas. This document defines a [MIME-DIR] usage profile for conveying directory information based on one such schema; that of the white-pages type of person object.

The schema is based on the attributes for the person object defined in the X.520 and X.521 directory services recommendations. The schema has augmented the basic attributes defined in the X.500 series recommendation in order to provide for an electronic representation of the information commonly found on a paper business card. This schema was first defined in the [VCARD] document. Hence, this [MIME- DIR] profile is referred to as the vCard MIME Directory Profile.

A directory entry based on this usage profile can include traditional directory, white-pages information such as the distinguished name used to uniquely identify the entry, a formatted representation of the name used for user-interface or presentation purposes, both the structured and presentation form of the delivery address, various telephone numbers and organizational information associated with the entry. In addition, traditional paper business card information such as an image of an organizational logo or identify photograph can be included in this person object.

The vCard MIME Directory Profile also provides support for representing other important information about the person associated with the directory entry. For instance, the date of birth of the person; an audio clip describing the pronunciation of the name associated with the directory entry, or some other application of the digital sound; longitude and latitude geo-positioning information related to the person associated with the directory entry; date and time that the directory information was last updated; annotations often written on a business card; Uniform Resource Locators (URL) for a website; public key information. The profile also provides support for

non-standard extensions to the schema. This provides the flexibility for implementations to augment the current capabilities of the profile in a standardized way.

RFC2460 - Internet Protocol, Version 6 (IPv6) Specification

return to the IETF Standards

Table 22: Data sheet for Transmission Control Protocol (TCP)

Title	Internet Protocol, Version 6 (IPv6) Specification
Acronym	IPv6
Version	1981
Document Number	RFC2460
Release Date	December 1998
Reference	https://tools.ietf.org/html/rfc2460

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

IP version 6 (IPv6) is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4) [RFC-791]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- **Expanded Addressing Capabilities**: IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a “scope” field to multicast addresses. And a new type of address called an “anycast address” is defined, used to send a packet to any one of a group of nodes.
- **Header Format Simplification**: Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.
- **Improved Support for Extensions and Options**: Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- **Flow Labeling Capability**: A new capability is added to enable the labeling of packets belonging to particular traffic “flows” for which the sender requests special handling, such as non-default quality of service or “real-time” service.
- **Authentication and Privacy Capabilities**: Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

This document specifies the basic IPv6 header and the initially- defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in [ADDRARCH]. The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [ICMPv6].

RFC2818 - HTTP Over TLS (HTTPS)

return to the IETF Standards

Table 23: Data sheet for HTTP Over TLS (HTTPS)

Title	HTTP Over TLS
Acronym	HTTPS
Version	
Document Number	8
Release Date	May 2000
Reference	https://tools.ietf.org/html/rfc2818

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

HTTP [RFC2616] was originally used in the clear on the Internet. However, increased use of HTTP for sensitive applications has required security measures. SSL, and its successor TLS RFC2246 were designed to provide channel-oriented security. This document describes how to use HTTP over TLS.

RFC3339 - Date and Time on the Internet: Timestamps

return to the IETF Standards

Table 24: Data sheet for Date and Time on the Internet: Timestamps

Title	Date and Time on the Internet: Timestamps
Acronym	Timestamps
Version	2002
Document Number	RFC3339
Release Date	July 2002
Reference	https://tools.ietf.org/html/rfc3339

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

Date and time formats cause a lot of confusion and interoperability problems on the Internet. This document addresses many of the problems encountered and makes recommendations to improve consistency and interoperability when representing and using date and time in Internet protocols.

This document includes an Internet profile of the ISO 8601 [ISO8601] standard for representation of dates and times using the Gregorian calendar.

There are many ways in which date and time values might appear in Internet protocols: this document focuses on just one common usage, viz. timestamps for Internet protocol events. This limited consideration has the following consequences:

- *All dates and times are assumed to be in the "current era", somewhere between 0000AD and 9999AD.*
- *All times expressed have a stated relationship (offset) to Coordinated Universal Time (UTC). (This is distinct from some usage in scheduling applications where a local time and location may be known, but the actual relationship to UTC may be dependent on the unknown or unknowable actions of politicians or administrators. The UTC time corresponding to 17:00 on 23rd March 2005 in New York may depend on administrative decisions about daylight savings time. This specification steers well clear of such considerations.)*
- *Timestamps can express times that occurred before the introduction of UTC. Such timestamps are expressed relative to universal time, using the best available practice at the stated time.*
- *Date and time expressions indicate an instant in time. Description of time periods, or intervals, is not covered here.*

RFC3447 - PKCS #1: RSA Cryptography Specifications

return to the IETF Standards

Table 25: Data sheet for PKCS #1: RSA Cryptography Specifications

Title	PKCS #1: RSA Cryptography Specifications
Acronym	PKCS-RSA
Version	version 2.1
Document Number	RFC3447
Release Date	February 2003
Reference	https://www.ietf.org/rfc/rfc3447.txt

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This document provides recommendations for the implementation of public-key cryptography based on the RSA algorithm [168], covering the following aspects:

- *Cryptographic primitives*
- *Encryption schemes*
- *Signature schemes with appendix*
- *ASN.1 syntax for representing keys and for identifying the schemes*

The recommendations are intended for general application within computer and communications systems, and as such include a fair amount of flexibility. It is expected that application standards based on these specifications may include additional constraints. The recommendations are intended to be compatible with the standard IEEE-1363-2000 [26] and draft standards currently being developed by the ANSI X9F1 [1] and IEEE P1363 [27] working groups.

168)
R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21 (2), pp. 120-126, February 1978.

RFC3596 - DNS Extension to support IP Version 6

return to the IETF Standards

Table 26: Data sheet for DNS Extension to support IP Version 6

Title	DNS Extension to support IP Version 6
Acronym	
Version	2003
Document Number	RFC3596
Release Date	October 2003
Reference	https://tools.ietf.org/html/rfc3596

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This document defines the changes that need to be made to the Domain Name System (DNS) to support hosts running IP version 6 (IPv6). The changes include a resource record type to store an IPv6 address, a domain to support lookups based on an IPv6 address, and updated definitions of existing query types that return Internet addresses as part of additional section processing. The extensions are designed to be compatible with existing applications and, in particular, DNS implementations themselves.

RFC4122 - A Universally Unique IDentifier (UUID) URN Namespace

return to the IETF Standards

Table 27: Data sheet for A Universally Unique IDentifier (UUID) URN Namespace

Title	A Universally Unique IDentifier (UUID) URN Namespace
Acronym	UUID
Version	2005
Document Number	RFC4122
Release Date	July 2005
Reference	https://tools.ietf.org/html/rfc4122

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This specification defines a Uniform Resource Name namespace for UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier). A UUID is 128 bits long, and can guarantee uniqueness across space and time. UUIDs were originally used in the Apollo Network Computing System and later in the Open Software Foundation's (OSF) Distributed Computing Environment (DCE), and then in Microsoft Windows platforms.

This specification is derived from the DCE specification with the kind permission of the OSF (now known as The Open Group). Information from earlier versions of the DCE specification have been incorporated into this document.

RFC5011 - Automated Updates of DNS Security (DNSSEC) Trust Anchors

return to the IETF Standards

Table 28: Data sheet for RFC5011 Automated Updates of DNS Security (DNSSEC) Trust Anchors (AAAA)

Title	Automated Updates of DNS Security Trust Anchors
Acronym	DNSSEC
Version	2007
Document Number	RFC5011
Release Date	September 2007
Reference	https://tools.ietf.org/html/rfc5011

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This document describes a means for automated, authenticated, and authorized updating of DNSSEC "trust anchors". The method provides protection against N-1 key compromises of N keys in the trust point key set. Based on the trust established by the presence of a current anchor, other anchors may be added at the same place in the hierarchy, and, ultimately, supplant the existing anchor(s).

This mechanism will require changes to resolver management behavior (but not resolver resolution behavior), and the addition of a single flag bit to the DNSKEY record.

RFC5424 - The Syslog Protocol (SYSLOG)

return to the IETF Standards

Table 29: Data sheet for The Syslog Protocol (SYSLOG)

Title	The Syslog Protocol
Acronym	SYSLOG
Version	2009
Document Number	RFC5424
Release Date	March 2009
Reference	https://tools.ietf.org/html/rfc5424

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This document describes a layered architecture for syslog. The goal of this architecture is to separate message content from message transport while enabling easy extensibility for each layer.

This document describes the standard format for syslog messages and outlines the concept of transport mappings. It also describes structured data elements, which can be used to transmit easily parseable, structured information, and allows for vendor extensions.

This document does not describe any storage format for syslog messages. It is beyond of the scope of the syslog protocol and is unnecessary for system interoperability.

This document has been written with the original design goals for traditional syslog in mind. The need for a new layered specification has arisen because standardization efforts for reliable and secure syslog extensions suffer from the lack of a Standards-Track and transport-independent RFC. Without this document, each other standard would need to define its own syslog packet format and transport mechanism, which over time will introduce subtle compatibility issues. This document tries to provide a foundation that syslog extensions can build on. This layered architecture approach also provides a solid basis that allows code to be written once for each syslog feature instead of once for each transport.

This document obsoletes RFC 3164, which is an Informational document describing some implementations found in the field.

RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0

return to the IETF Standards

Table 30: Data sheet for The Secure Sockets Layer (SSL) Protocol Version 3.0

Title	The Secure Sockets Layer (SSL) Protocol Version 3.0
Acronym	SSL
Version	3.0
Document Number	RFC6101
Release Date	August 2011
Reference	https://tools.ietf.org/html/rfc6101

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The primary goal of the SSL protocol is to provide privacy and reliability between two communicating applications. The protocol is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP RFC0793), is the SSL record protocol. The SSL record protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the SSL handshake protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent. A higher level protocol can layer on top of the SSL protocol transparently. The SSL protocol provides connection security that has three basic properties:

- *The connection is private. Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (e.g., DES, 3DES, RC4).*
- *The peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA, DSS).*
- *The connection is reliable. Message transport includes a message integrity check using a keyed Message Authentication Code (MAC) [RFC2104]. Secure hash functions (e.g., SHA, MD5) are used for MAC computations.*

RFC6376 - DomainKeys Identified Mail (DKIM) Signatures

return to the IETF Standards

Table 31: Data sheet for RFC6376 DomainKeys Identified Mail (DKIM) Signatures (AAAA)

Title	DomainKeys Identified Mail Signatures
Acronym	DKIM
Version	2011
Document Number	RFC6376
Release Date	September 2011
Reference	https://tools.ietf.org/html/rfc6376

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

DomainKeys Identified Mail (DKIM) permits a person, role, or organization that owns the signing domain to claim some responsibility for a message by associating the domain with the message. This can be an author's organization, an operational relay, or one of their agents. DKIM separates the question of the identity of the Signer of the message from the purported author of the message. Assertion of responsibility is validated through a cryptographic signature and by querying the Signer's domain directly to retrieve the appropriate public key. Message transit from author to recipient is through relays that typically make no substantive change to the message content and thus preserve the DKIM signature.

RFC6455 - The WebSocket Protocol

return to the IETF Standards

Table 32: Data sheet for RFC6455 The WebSocket Protocol

Title	The WebSocket Protocol
Acronym	
Version	2011
Document Number	RFC6455
Release Date	December 2011
Reference	https://tools.ietf.org/html/rfc6455

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).

RFC6749 - The OAuth 2.0 Authorization Framework

return to the IETF Standards

Table 33: Data sheet for The OAuth 2.0 Authorization Framework

Title	Transmission Control Protocol
Acronym	OAuth
Version	2.0
Document Number	RFC6749
Release Date	October 2012
Reference	https://tools.ietf.org/html/rfc6749

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

In the traditional client-server authentication model, the client requests an access-restricted resource (protected resource) on the server by authenticating with the server using the resource owner's credentials. In order to provide third-party applications access to restricted resources, the resource owner shares its credentials with the third party. This creates several problems and limitations:

- *Third-party applications are required to store the resource owner's credentials for future use, typically a password in clear-text.*
- *Servers are required to support password authentication, despite the security weaknesses inherent in passwords.*
- *Third-party applications gain overly broad access to the resource owner's protected resources, leaving resource owners without any ability to restrict duration or access to a limited subset of resources.*
- *Resource owners cannot revoke access to an individual third party without revoking access to all third parties, and must do so by changing the third party's password.*
- *Compromise of any third-party application results in compromise of the end-user's password and all of the data protected by that password.*

OAuth addresses these issues by introducing an authorization layer and separating the role of the client from that of the resource owner. In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource owner.

Instead of using the resource owner's credentials to access protected resources, the client obtains an access token – a string denoting a specific scope, lifetime, and other access attributes. Access

tokens are issued to third-party clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server.

For example, an end-user (resource owner) can grant a printing service (client) access to her protected photos stored at a photo- sharing service (resource server), without sharing her username and password with the printing service. Instead, she authenticates directly with a server trusted by the photo-sharing service (authorization server), which issues the printing service delegation- specific credentials (access token).

This specification is designed for use with HTTP (RFC2616). The use of OAuth over any protocol other than HTTP is out of scope.

The OAuth 1.0 protocol (RFC5849), published as an informational document, was the result of a small ad hoc community effort. This Standards Track specification builds on the OAuth 1.0 deployment experience, as well as additional use cases and extensibility requirements gathered from the wider IETF community. The OAuth 2.0 protocol is not backward compatible with OAuth 1.0. The two versions may co-exist on the network, and implementations may choose to support both. However, it is the intention of this specification that new implementations support OAuth 2.0 as specified in this document and that OAuth 1.0 is used only to support existing deployments. The OAuth 2.0 protocol shares very few implementation details with the OAuth 1.0 protocol. Implementers familiar with OAuth 1.0 should approach this document without any assumptions as to its structure and details.

RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage

return to the IETF Standards

Table 34: The OAuth 2.0 Authorization Framework: Bearer Token Usage

Title	The OAuth 2.0 Authorization Framework: Bearer Token Usage
Acronym	TCP
Version	2.0
Document Number	RFC6750
Release Date	October 2012
Reference	https://tools.ietf.org/html/rfc6750

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

OAuth enables clients to access protected resources by obtaining an access token, which is defined in "The OAuth 2.0 Authorization Framework" [RFC6749] as "a string representing an access authorization issued to the client", rather than using the resource owner's credentials directly.

Tokens are issued to clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server. This specification describes how to make protected resource requests when the OAuth access token is a bearer token.

This specification defines the use of bearer tokens over HTTP/1.1 RFC2616 using Transport Layer Security (TLS) RFC5246 to access protected resources. TLS is mandatory to implement and use with this specification; other specifications may extend this specification for use with other protocols. While designed for use with access tokens resulting from OAuth 2.0 authorization RFC6749 flows to access OAuth protected resources, this specification actually defines a general HTTP authorization method that can be used with bearer tokens from any source to access any resources protected by those bearer tokens. The Bearer authentication scheme is intended primarily for server authentication using the WWW-Authenticate and Authorization HTTP headers but does not preclude its use for proxy authentication.

RFC6891 - Extension Mechanisms for DNS (EDNS(0))

return to the IETF Standards

Table 35: Data sheet for RFC6891 Extension Mechanisms for DNS (EDNS(0))

Title	Extension Mechanisms for DNS
Acronym	EDNS(0)
Version	2013
Document Number	RFC6891
Release Date	April 2013
Reference	https://tools.ietf.org/html/rfc6891

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The Domain Name System's wire protocol includes a number of fixed fields whose range has been or soon will be exhausted and does not allow requestors to advertise their capabilities to responders. This document describes backward-compatible mechanisms for allowing the protocol to grow.

This document updates the Extension Mechanisms for Domain Name System (DNS) (EDNS(0)) specification (and obsoletes RFC 2671) based on feedback from deployment experience in several implementations. It also obsoletes RFC 2673 (“Binary Labels in the Domain Name System”) and adds considerations on the use of extended labels in the DNS.

RFC6979 - Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)

return to the IETF Standards

Table 36: Data sheet for Transmission Control Protocol (TCP)

Title	Transmission Control Protocol
Acronym	ECDSA
Version	2013
Document Number	RFC6979
Release Date	August 2013
Reference	https://tools.ietf.org/html/rfc6979

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This document defines a deterministic digital signature generation procedure. Such signatures are compatible with standard Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) digital signatures and can be processed with unmodified verifiers, which need not be aware of the procedure described therein. Deterministic signatures retain the cryptographic security features associated with digital signatures but can be more easily implemented in various environments, since they do not need access to a source of high-quality randomness.

RFC7061 - eXtensible Access Control Markup Language (XACML) XML Media Type

return to the IETF Standards

Table 37: Data sheet for eXtensible Access Control Markup Language (XACML) XML Media Type

Title	eXtensible Access Control Markup Language (XACML) XML Media Type
Acronym	XACML
Version	1981
Document Number	RFC7061
Release Date	November 2013 1981
Reference	https://tools.ietf.org/html/rfc7061

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The eXtensible Access Control Markup Language (XACML) [XACML-3] defines an architecture and a language for access control (authorization). The language consists of requests, responses, and policies. Clients send a request to a server to query whether a given action should be allowed. The server evaluates the request against the available policies and returns a response. The policies implement the organization's access control requirements.

RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication

return to the IETF Standards

Table 38: Data sheet for Hypertext Transfer Protocol (HTTP/1.1): Authentication

Title	Transmission Control Protocol
Acronym	HTTP
Version	1.1
Document Number	RFC7235
Release Date	June 2014
Reference	https://tools.ietf.org/html/rfc7235

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

HTTP provides a general framework for access control and authentication, via an extensible set of challenge-response authentication schemes, which can be used by a server to challenge a client request and by a client to provide authentication information. This document defines HTTP/1.1 authentication in terms of the architecture defined in "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing" RFC7230, including the general framework previously described in "HTTP Authentication: Basic and Digest Access Authentication" RFC2617 and the related fields and status codes previously defined in "Hypertext Transfer Protocol – HTTP/1.1" RFC2616.

The IANA Authentication Scheme Registry (Section 5.1) lists registered authentication schemes and their corresponding specifications, including the "basic" and "digest" authentication schemes previously defined by RFC 2617.

RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format

return to the IETF Standards

Table 39: Data sheet for The JavaScript Object Notation (JSON) Data Interchange Format

Title	The JavaScript Object Notation (JSON) Data Interchange Format
Acronym	JSON
Version	2017
Document Number	RFC8259
Release Date	December 2017
Reference	https://tools.ietf.org/html/rfc8259

Note: The following is an excerpt from the official IETF RFC. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format. It was derived from the ECMAScript Programming Language Standard. JSON defines a small set of formatting rules for the portable representation of structured data.

This document removes inconsistencies with other specifications of JSON, repairs specification errors, and offers experience-based interoperability guidance.

International Organization for Standardization (ISO)

return to the Technical Standards area

Create a New Page for ISO/IEC **Number** →		Add page

ISO is an independent, non-governmental international organization with a membership of 164 national standards bodies.

> *Through its members, it brings together experts to share knowledge and develop voluntary, consensus-based, market relevant International Standards that support innovation and provide solutions to global challenges. All about ISO*

Technical Standards

- ISO/IEC 19506:2012 Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM)
- ISO/IEC 23360-1:2006 Linux Standard Base (LSB) core specification 3.1 -- Part 1: Generic specification
- ISO 9001:2015 Quality management
- ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
- ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE
- ISO/IEC 25001:2014 SQuaRE -- Planning and Management
- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- ISO/IEC 25012:2008 SQuaRE -- Data Quality Model
- ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide
- ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements
- ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality
- ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality
- ISO/IEC 25030:2007 SQuaRE -- Quality Requirements
- ISO/IEC 25040:2011 SQuaRE -- Evaluation Process
- ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators
- ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability
- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ISO/IEC 22275:2018 Programming Languages - ECMAScript Specification Suite
- ISO 8601-1:2019 Date and time -- Representations for information interchange -- Part 1: Basic rules
- ISO 8601-2:2019 Date and time -- Representations for information interchange -- Part 2: Extensions: Basic rules
- ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes
- ISO/IEC 9834-8:2014 Information technology -- Procedures for the operation of object identifier

registration authorities -- Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers

ISO/IEC 19506:2012 Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM)

return to the ISO Standards

Table 40: Data sheet for Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)

Title	Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)
Version	2012
Document Number	19506:2012
Release Date	2012-04
Reference	https://www.iso.org/standard/32625.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 19506:2012 defines a meta-model for representing existing software assets, their associations, and operational environments, referred to as the Knowledge Discovery Meta-model (KDM). This is the first in the series of specifications related to Software Assurance (SwA) and Architecture-Driven Modernization (ADM) activities. KDM facilitates projects that involve existing software systems by insuring interoperability and exchange of data between tools provided by different vendors.

ISO/IEC 23360-1:2006 Linux Standard Base (LSB) core specification 3.1 -- Part 1: Generic specification

return to the ISO Standards

Table 41: Data sheet for Linux Standard Base (LSB) core specification 3.1 – Part 1: Generic specification

Title	Linux Standard Base (LSB) core specification 3.1 – Part 1: Generic specification
Acronym	LSB
Version	2012
Document Number	23360-1:2006
Release Date	2006-12
Reference	https://www.iso.org/standard/43781.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

The Linux Standard Base (LSB) defines a system interface for compiled applications and a minimal environment for support of installation scripts. Its purpose is to enable a uniform industry standard environment for high-volume applications conforming to the LSB.

These specifications are composed of two basic parts: A common specification ("LSB-generic" or "generic LSB"), ISO/IEC 23360-1:2006, describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part ("LSB-arch" or "archLSB") describing the parts of the interface that vary by processor architecture. Together, the LSB-generic and the relevant architecture-specific parts of ISO/IEC 23360 for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture.

ISO/IEC 23360-1:2006, the LSB-generic document, is used in conjunction with an architecture-specific part. Whenever a section of the LSB-generic specification is supplemented by architecture-specific information, the LSB-generic document includes a reference to the architecture part.

Note ISO/IEC 23360-1:2006 is undergoing revisions. It will be replaced by the following:

- ISO/IEC 23360-1:2006 Linux Standard Base (LSB) core specification 3.1 – Part 1: Generic specification
- ISO/IEC 23360-2:2006 Linux Standard Base (LSB) core specification 3.1 – Part 2: Specification for IA-32 architecture
- ISO/IEC 23360-3:2006 Linux Standard Base (LSB) core specification 3.1 – Part 3: Specification for IA-64 architecture

- ISO/IEC 23360-4:2006 Linux Standard Base (LSB) core specification 3.1 – Part 4: Specification for AMD64 architecture
- ISO/IEC 23360-5:2006 Linux Standard Base (LSB) core specification 3.1 – Part 5: Specification for PPC32 architecture
- ISO/IEC 23360-6:2006 Linux Standard Base (LSB) core specification 3.1 – Part 6: Specification for PPC64 architecture
- ISO/IEC 23360-7:2006 Linux Standard Base (LSB) core specification 3.1 – Part 7: Specification for S390 architecture
- ISO/IEC 23360-8:2006 Linux Standard Base (LSB) core specification 3.1 – Part 8: Specification for S390X architecture

ISO 9001:2015 Quality management

return to the ISO Standards

Table 42: Data Sheet for Quality management

Title	Quality management
Acronym	ISO 900 Family
Version	2015
Document Number	9001:2015
Release Date	2015
Reference	https://www.iso.org/iso-9001-quality-management.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO 9001:2015 sets out the criteria for a quality management system and is the only standard in the family that can be certified to (although this is not a requirement). It can be used by any organization, large or small, regardless of its field of activity. In fact, there are over one million companies and organizations in over 170 countries certified to ISO 9001.

This standard is based on a number of quality management principles including a strong customer focus, the motivation and implication of top management, the process approach and continual improvement. These principles are explained in more detail in the pdf Quality Management Principles. Using ISO 9001:2015 helps ensure that customers get consistent, good quality products and services, which in turn brings many business benefits.

ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software

return to the ISO Standards

Table 43: Data Sheet for Software engineering – Guidelines for the application of ISO 9001:2015 to computer software

Title	Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
Acronym	
Version	2018
Document Number	90003:2018
Release Date	2018-11
Reference	https://www.iso.org/standard/63711.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

This document provides guidance for organizations in the application of ISO 9001:2015 to the acquisition, supply, development, operation and maintenance of computer software and related support services. It does not add to or otherwise change the requirements of ISO 9001:2015.

Annex A provides a table pointing to additional guidance on the implementation of ISO 9001:2015, available in ISO/IEC JTC 1/SC 7, ISO/IEC JTC 1/SC 27 and ISO/TC 176 International Standards.

The guidelines provided in this document are not intended to be used as assessment criteria in quality management system registration/certification. However, some organizations can consider it useful to implement the guidelines proposed in this document and can be interested in knowing whether the resultant quality management system is compliant or not with this document. In this case, an organization can use both this document and ISO 9001 as assessment criteria for quality management systems in the software domain.

ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE

return to the ISO Standards

Table 44: Data Sheet for Guide to SQuaRE

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE and software engineering – System life cycle processes
Acronym	SQuaRE
Version	2014
Document Number	25000:2014
Release Date	2014-03
Reference	https://www.iso.org/standard/64764.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25000:2014 provides guidance for the use of the new series of International Standards named Systems and software Quality Requirements and Evaluation (SQuaRE). The purpose of ISO/IEC 25000:2014 is to provide a general overview of SQuaRE contents, common reference models and definitions, as well as the relationship among the documents, allowing users of the Guide a good understanding of those series of standards, according to their purpose of use. It also contains an explanation of the transition process between the old ISO/IEC 9126 and the ISO/IEC 14598 series and SQuaRE.

ISO/IEC 25001:2014 SQuaRE -- Planning and Management

return to the ISO Standards

Table 45: Data Sheet for SQuaRE – Planning and Management

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation – Planning and management
Acronym	SQuaRE
Version	2014
Document Number	25001:2014
Release Date	2014-35
Reference	https://www.iso.org/standard/64787.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25001:2014 provides requirements and recommendations for an organization responsible for implementing and managing the systems and software product quality requirements specification and evaluation activities through the provision of technology, tools, experiences, and management skills.

The role of the evaluation group includes motivating employees and training them for the requirements specification and the evaluation activities, preparing appropriate documents, identification or development of required methods, and responding to queries on relevant technologies.

Technology management is related to the planning and management of a systems and software quality requirements specification and evaluation process, measurements and tools. This includes the management of development, acquisition, standardisation, control, transfer and feedback of requirements specification and evaluation technology experiences within the organisation.

The intended users of ISO/IEC 25001:2014 are those responsible for:

- *managing technologies used for requirements specification and evaluation execution,*
- *mspecifying systems and software product quality requirements,*
- *msupporting systems and software product quality evaluation,*
- *mmanaging systems and software development organisations,*

as well as those in a quality assurance function. However, it is also applicable to managers involved in other systems or software related activities.

ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models

return to the ISO Standards

Table 46: Data Sheet for SQuaRE – System and Software Quality Models

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models
Acronym	
Version	2011
Document Number	2510:2011
Release Date	2011-03
Reference	https://www.iso.org/standard/35733.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25010:2011 defines:

1. *A quality in use model composed of five characteristics (some of which are further subdivided into subcharacteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use.*
2. *A product quality model composed of eight characteristics (which are further subdivided into subcharacteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products.*

The characteristics defined by both models are relevant to all software products and computer systems. The characteristics and subcharacteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness.

Although the scope of the product quality model is intended to be software and computer systems, many of the characteristics are also relevant to wider systems and services.

ISO/IEC 25012 contains a model for data quality that is complementary to this model.

The scope of the models excludes purely functional properties, but it does include functional suitability.
The scope of application of the quality models includes supporting specification and evaluation of

software and software-intensive computer systems from different perspectives by those associated with their acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and control, and audit. The models can, for example, be used by developers, acquirers, quality assurance and control staff and independent evaluators, particularly those responsible for specifying and evaluating software product quality. Activities during product development that can benefit from the use of the quality models include:

- *identifying software and system requirements;*
- *validating the comprehensiveness of a requirements definition;*
- *identifying software and system design objectives;*
- *identifying software and system testing objectives;*
- *identifying quality control criteria as part of quality assurance;*
- *identifying acceptance criteria for a software product and/or software-intensive computer system;*
- *establishing measures of quality characteristics in support of these activities.*

ISO/IEC 25012:2008 SQuaRE -- Data Quality Model

return to the ISO Standards

Table 47: Data Sheet for SQuaRE – Data Quality Model

Title	Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model
Acronym	
Version	2008
Document Number	25012:2008
Release Date	2008-12
Reference	https://www.iso.org/standard/35736.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25012:2008 defines a general data quality model for data retained in a structured format within a computer system.

ISO/IEC 25012:2008 can be used to establish data quality requirements, define data quality measures, or plan and perform data quality evaluations. It could be used, for example,

- *to define and evaluate data quality requirements in data production, acquisition and integration processes,*
- *to identify data quality assurance criteria, also useful for re-engineering, assessment and improvement of data,*
- *to evaluate the compliance of data with legislation and/or requirements.*
- *ISO/IEC 25012:2008 categorizes quality attributes into fifteen characteristics considered by two points of view: inherent and system dependent. Data quality characteristics will be of varying importance and priority to different stakeholders.*

ISO/IEC 25012:2008 is intended to be used in conjunction with the other parts of the SQuaRE series of International Standards, and with ISO/IEC 9126-1 until superseded by ISO/IEC 25010.

ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide

return to the ISO Standards

Table 48: Data Sheet for SQuaRE – Measurement Reference Model and Guide

Title	Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement reference model and guide
Acronym	
Version	2007
Document Number	25020:2007
Release Date	2007-05
Reference	https://www.iso.org/standard/35744.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25020:2007 provides a measurement reference model and guide for measuring the quality characteristics defined in ISO/IEC 2501n, Quality Model Division. ISO/IEC 25020:2007 sets requirements for the selection and construction of quality measures. It also contains informative annexes addressing the following topics: criteria for selecting software quality measures and quality measure elements, demonstrating predictive validity and assessing measurement reliability, and an example format for documenting software quality measures.

The Quality Measurement Division, of which ISO/IEC 25020 is a member, also offers examples of quality measures that can be used across the product development life cycle. These measures are defined in the other documents in the division and correspond to the quality characteristics in a software product quality model such as that described in ISO/IEC 25010. ISO/IEC 25020:2007 and the quality measures are designed to be used, in particular, with other standards in the SQuaRE series that address quality requirements (ISO/IEC 25030) and product quality evaluation (ISO/IEC 25040).

ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements

return to the ISO Standards

Table 49: Data Sheet for SQuaRE Quality Measure Elements

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measure elements
Acronym	
Version	2012
Document Number	25021:2012
Release Date	2012-1
Reference	https://www.iso.org/standard/55477.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25021:2012 provides guides to specify Quality Measure Elements (QME) and initial set of QME as examples. QME is a measure defined in terms of a property and the measurement method for quantifying it, including optionally the transformation by a mathematical function.

ISO/IEC 25021:2012 is intended to be used throughout the system and software product life cycle used with other documents of the ISO/IEC 250nn SQuaRE series, especially ISO/IEC 25022, ISO/IEC 25023 and ISO/IEC 25024.

ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use

return to the ISO Standards

Table 50: Data Sheet for SQuaRE – Measurement of Quality in Use

Title	Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Measurement of quality in use
Acronym	
Version	2016
Document Number	25022:2016
Release Date	2016-06
Reference	https://www.iso.org/standard/35746.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25022:2016 defines quality in use measures for the characteristics defined in ISO/IEC 25010, and is intended to be used together with ISO/IEC 25010. It can be used in conjunction with the ISO/IEC 2503n and the ISO/IEC 2504n standards or to more generally meet user needs with regard to product or system quality.

ISO/IEC 25022:2016 contains the following:

- *a basic set of measures for each quality in use characteristic;*
- *an explanation of how quality in use is measured.*

It provides a suggested set of quality in use measures to be used with the quality in use model in ISO/IEC 25010. They are not intended to be an exhaustive set.

It includes as informative annexes examples of how to measure context coverage (Annex A), options for normalising quality in use measures (Annex B), use of ISO/IEC 25022 for measuring usability in ISO 9241-11 (Annex C), a quality in use evaluation process (Annex D), the relationship between different quality models (Annex E), and quality measurement concepts (Annex F).

The measures are applicable to the use of any human-computer system, including both computer systems in use and software products that form part of the system.

It does not assign ranges of values of the measures to rated levels or to grades of compliance because these values are defined for each system or product depending, on the context of use and users' needs.

Some attributes could have a desirable range of values, which does not depend on specific user needs but depends on generic factors, for example, human cognitive factors.

The proposed quality in use measures are primarily intended to be used for quality assurance and management of systems and software products based on their effects when actually used. The main users of the measurement results are people managing development, acquisition, evaluation, or maintenance of software and systems.

The main users of ISO/IEC 25022:2016 are people carrying out specification and evaluation activities as part of the following:

- *development: including requirements analysis, design, and testing through acceptance during the life cycle process;*
- *quality management: systematic examination of the product or computer system, for example, when evaluating quality in use as part of quality assurance and quality control;*
- *supply: a contract with the acquirer for the supply of a system, software product, or software service under the terms of a contract, for example, when validating quality at qualification test;*
- *acquisition: including product selection and acceptance testing, when acquiring or procuring a system, software product, or software service from a supplier;*
- *maintenance: improvement of the product based on quality in use measures.*

ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality

return to the ISO Standards

Table 51: Data Sheet for SQuaRE – Measurement of System and Software Product Quality

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality
Acronym	
Version	2016
Document Number	25023:2016
Release Date	2016-06
Reference	https://www.iso.org/standard/35747.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25023:2016 defines quality measures for quantitatively evaluating system and software product quality in terms of characteristics and subcharacteristics defined in ISO/IEC 25010 and is intended to be used together with ISO/IEC 25010. It can be used in conjunction with the ISO/IEC 2503n and the ISO/IEC 2504n standards or to more generally meet user needs with regard to software product or system quality.

ISO/IEC 25023:2016 contains the following:

- *a basic set of quality measures for each characteristic and subcharacteristics;*
- *an explanation of how to apply software product and system quality measures.*

It includes, as informative annexes, considerations for the use of quality measures (Annex A), QMEs used to define product or system quality measures (Annex B), and detailed explanation of measurement types (Annex C). : ISO/IEC 25023:2016 does not assign ranges of values of the measures to rated levels or to grades of compliance because these values are defined based on the nature of the system, product or a part of the product, and depending on factors such as category of the software, integrity level, and users' needs. Some attributes could have a desirable range of values, which does not depend on specific user needs but depends on generic factors; for example, human cognitive factors. *:* The proposed quality measures are primarily intended to be used for quality assurance and improvement of system and software products during or post the development life cycle process. *:* The main users of ISO/IEC 25023:2016 are people carrying out quality requirement specification and evaluation activities as part of the following: * development: including requirements analysis, design specification, coding and testing through acceptance

during the life cycle process; * quality management: systematic examination of the software product or computer system, for example, when evaluating system or software product quality as part of quality assurance, quality control and quality certification; * supply: a contract with the acquirer for the supply of a system, software product or software service under the terms of a contract, for example, when validating quality at qualification test; * acquisition: including product selection and acceptance testing, when acquiring or procuring a system, software product or software service from a supplier; * maintenance: improvement of the software product or system based on quality measurement.

ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality

return to the ISO Standards

Table 52: Data Sheet for SQuaRE – Measurement of Data Quality

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of data quality
Acronym	
Version	2015
Document Number	25024:2015
Release Date	2015-10
Reference	https://www.iso.org/standard/35749.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25024:2015 defines data quality measures for quantitatively measuring the data quality in terms of characteristics defined in ISO/IEC 25012.

ISO/IEC 25024:2015 contains the following:

- *a basic set of data quality measures for each characteristic;*
- *a basic set of target entities to which the quality measures are applied during the data-life-cycle;*
- *an explanation of how to apply data quality measures;*
- *a guidance for organizations defining their own measures for data quality requirements and evaluation.*

It includes, as informative annexes, a synoptic table of quality measure elements defined in this International standard (Annex A), a table of quality measures associated to each quality measure element and target entitiy (Annex B), considerations about specific quality measure elements (Annex C), a list of quality measures in alphabetic order (Annex D), and a table of quality measures grouped by characteristics and target entities (Annex E).

This International Standard does not define ranges of values of these quality measures to rate levels or grades because these values are defined for each system by its nature depending on the system context and users' needs.

This International Standard can be applied to any kind of data retained in a structured format within a computer system used for any kinds of applications.

People managing data and services including data are the primary beneficiaries of the quality measures.

This International Standard is intended to be used by people who need to produce and/or use data quality measures while pursuing their responsibilities.

- *Acquirer (an individual or organization that acquires or procures data from a supplier).*
- *Evaluator (an individual or organization that performs an evaluation, which can, for example, be a testing laboratory, the quality department of an organization, a government organization, or a user).*
- *Developer (an individual or organization that performs development activities including requirements, analysis, design, implementation, and testing data during the data-life-cycle).*
- *Maintainer (an individual or organization that performs operation and maintenance activities of data).*
- *Supplier (an individual or organization that enters into a contract with the acquirer for the supply of data or service under the terms of the contract).*
- *User (an individual or organization that uses data to perform a specific function).*
- *Quality manager (an individual or organization that performs a systematic examination of the data).*
- *Owner (an individual or organization that takes responsibility for the management and financial value of the data with the legal authority and responsibility to establish for them evaluation, collections, access, dissemination, storage, security, and cancellation).*

ISO/IEC 25024:2015 takes into account a large range of data of target entities.

It can be applied in many types of information systems, for example, such as follows:

- *legacy information system;*
- *data warehouse;*
- *distributed information system;*
- *cooperative information system;*
- *world wide web.*

The scope does not include the following:

- *knowledge representation;*
- *data mining techniques;*
- *statistical significance for random sample.*

ISO/IEC 25030:2007 SQuaRE -- Quality Requirements

return to the ISO Standards

Table 53: Data Sheet for SQuaRE – Quality Requirements

Title	Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality requirements
Acronym	
Version	2007
Document Number	25030:2007
Release Date	2007-06
Reference	https://www.iso.org/standard/35755.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25030:2007 provides requirements and recommendations for the specification of software quality requirements. It applies to both acquirers and suppliers. It focuses on software quality requirements, but takes a system perspective since software is normally developed and applied as part of a larger system.

Software product quality requirements are needed for:

- *specification (including contractual agreement and call for tender);*
- *planning (including feasibility analysis);*
- *development (including early identification of potential quality problems during development); and*
- *evaluation (including objective assessment and certification of software product quality).*

If software quality requirements are not stated clearly, they may be viewed, interpreted, implemented and evaluated differently by different people. This may result in: software which is inconsistent with user expectations and of poor quality; users, clients and developers who are unsatisfied; and time and cost overruns to rework software.

ISO/IEC 25030:2007 helps to improve the quality of software quality requirements. It does this by providing requirements and recommendations for quality requirements, and guidance for the processes used to define and analyse quality requirements. It applies the quality model defined in ISO/IEC 9126-1 [ISO/IEC 25010] and it complies with the requirement processes defined in ISO/IEC 15288.

ISO/IEC 25030 is part of the SQuaRE series of International Standards.

ISO/IEC 25040:2011 SQuaRE -- Evaluation Process

return to the ISO Standards

Table 54: Data Sheet for SQuaRE – Evaluation Process

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process
Acronym	
Version	2011
Document Number	25040:2011
Release Date	2011-03
Reference	https://www.iso.org/standard/35765.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25040:2011 contains requirements and recommendations for the evaluation of software product quality and clarifies the general concepts. It provides a process description for evaluating software product quality and states the requirements for the application of this process. The evaluation process can be used for different purposes and approaches. The process can be used for the evaluation of the quality of pre-developed software, commercial-off-the-shelf software or custom software and can be used during or after the development process.

ISO/IEC 25040:2011 establishes the relationship of the evaluation reference model to the SQuaRE documents as well as shows how each SQuaRE document should be used during the activities of the evaluation process.

It is intended for those responsible for software product evaluation and is appropriate for developers, acquirers and independent evaluators of software products. These three different approaches are detailed in ISO/IEC 14598-3, ISO/IEC 14598-4, and ISO/IEC 14598-5.

It is not intended for evaluation of other aspects of software products (such as functional requirements, process requirements, business requirements, etc.).

ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators

return to the ISO Standards

Table 55: Data Sheet for SQuaRE – Evaluation Guide for Developers, Acquirers and Independent Evaluators

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation guide for developers, acquirers and independent evaluators
Acronym	
Version	2012
Document Number	25041:2012
Release Date	2012-10
Reference	https://www.iso.org/standard/35766.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25041:2012 provides requirements, recommendations and guidelines for system and software product quality evaluation, for the application of ISO/IEC 25040. Intended audiences of ISO/IEC 25041:2012 are developers, acquirers and independent evaluators of the system and software product. ISO/IEC 25041:2012 is part of ISO/IEC 250nn SQuaRE series of International Standards. It is not limited to any specific application area, and can be used for quality evaluation of any type of system and software product.

ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability

return to the ISO Standards

Table 56: Data Sheet for SQuaRE – Evaluation Module for Recoverability

Title	Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation module for recoverability
Acronym	
Version	2010
Document Number	25045:2010
Release Date	2010-09
Reference	https://www.iso.org/standard/35770.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 25045:2010 is one of the SQuaRE series of International Standards, which provides a framework for software products quality requirements and evaluation including the requirements for methods of software product measurement and evaluation.

ISO/IEC 25045:2010 uses a methodology involving two types of evaluation for recoverability. One part of the method makes use of the disturbance injection methodology and a list of disturbances based on common categories of operational faults and events to evaluate the quality measure of resiliency. The second quality measure is based on a set of questions that is defined for each disturbance to evaluate the quality measure of autonomic recovery index by assessing how well the system detects, analyses, and resolves the disturbance without human intervention.

ISO/IEC 25045:2010 is applicable to information systems executing transactions in a system supporting single or multiple concurrent users, where speedy recovery and ease of managing recovery is important to the acquirer, owner/operator, and the developer.

ISO/IEC 9899:2018 Programming languages -- C

return to the ISO Standards

Table 57: Data sheet for Programming languages – C

Title	Programming languages – C
Acronym	C
Version	2018
Document Number	9899:2018
Release Date	2018-06
Reference	https://www.iso.org/standard/74528.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

This document specifies the form and establishes the interpretation of programs written in the C programming language.
It specifies:

1. *the representation of C programs;*
2. *the syntax and constraints of the C language;*
3. *the semantic rules for interpreting C programs*
4. *the representation of input data to be processed by C programs;*
5. *the representation of output data produced by C programs;*
6. *the restrictions and limits imposed by a conforming implementation of C.*

This document does not specify:

1. *the mechanism by which C programs are transformed for use by a data-processing system;*
2. *the mechanism by which C programs are invoked for use by a data-processing system;*
3. *the mechanism by which input data are transformed for use by a C program;*
4. *the mechanism by which output data are transformed after being produced by a C program;*
5. *the size or complexity of a program and its data that will exceed the capacity of any specific data-processing system or the capacity of a particular processor;*
6. *all minimal requirements of a data-processing system that is capable of supporting a conforming implementation.*

ISO/IEC 14882:2017 Programming languages -- C++

return to the ISO Standards

Table 58: Data sheet for Programming languages – C++

Title	Programming languages – C++
Acronym	C++
Version	5
Document Number	14882:2017
Release Date	2017-12
Reference	https://www.iso.org/standard/68564.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC 14882:2017 specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements and relaxations of the first requirement appear at various places within this document.

C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2011 Programming languages ? C (hereinafter referred to as the C standard). In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

ISO/IEC 22275:2018 Programming Languages - ECMAScript Specification Suite

return to the ISO Standards

Table 59: Data sheet for Programming Languages - ECMAScript Specification Suite

Title	Programming Languages - ECMAScript Specification Suite
Acronym	
Version	1
Document Number	22275:2018
Release Date	2018-05
Reference	https://www.iso.org/standard/73002.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

This International Standard defines the ECMAScript Specification Suite containing the ECMAScript programming language and its required and optional built-in libraries. It defines all the necessary components (both normative and informative) that is needed to implement this suite of standards. This suite does not change if one or more components are updated by a new standard edition. The Suite changes only when new components are added and / or old components are removed from it.

ISO 8601-1:2019 Date and time -- Representations for information interchange -- Part 1: Basic rules

return to the ISO Standards

Table 60: Data sheet for Date and time – Representations for information interchange – Part 1: Basic rules

Title	Date and time – Representations for information interchange – Part 1: Basic rules
Acronym	
Version	2019
Document Number	8601-1:2019
Release Date	2019-02
Reference	https://www.iso.org/standard/70907.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

This document specifies representations of dates of the Gregorian calendar and times based on the 24-hour clock, as well as composite elements of them, as character strings for use in information interchange. It is also applicable for representing times and time shifts based on Coordinated Universal Time (UTC).

This document excludes the representation of date elements from non-Gregorian calendars or times not from the 24-hour clock. This document does not address character encoding of representations specified in this document.

ISO 8601-2:2019 Date and time -- Representations for information interchange -- Part 2: Extensions: Basic rules

return to the ISO Standards

Table 61: Data sheet for Date and time – Representations for information interchange – Part 2: Extensions

Title	Date and time – Representations for information interchange – Part 2: Extensions
Acronym	
Version	2019
Document Number	8601-2:2019
Release Date	2019-02
Reference	hhttps://www.iso.org/standard/70908.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

This document specifies additional representations of dates of the Gregorian calendar and times based on the 24-hour clock that extend the basic rules and composite elements of those defined in ISO 8601-1. These representations are specified as character strings for use in information interchange. It is also applicable for representing times and time shifts based on Coordinated Universal Time (UTC).

These extensions include:

- *uncertain or approximate dates, or dates with portions unspecified;*
- *extended time intervals;*
- *divisions of a year;*
- *sets and choices of calendar dates;*
- *grouped time scale units;*
- *repeat rules for recurring time intervals; and*
- *date and time arithmetic.*

This document excludes the representation of date elements from non-Gregorian calendars, or times not from the 24-hour clock. This document does not address character encoding of representations specified in this document.

ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes

return to the ISO Standards

Table 62: Data Sheet for Systems and software engineering – System life cycle processes

Title	Systems and software engineering – System life cycle processes
Version	2015
Document Number	15288:2015
Release Date	2015-05
Reference	https://www.iso.org/standard/63711.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

ISO/IEC/IEEE 15288:2015 establishes a common framework of process descriptions for describing the life cycle of systems created by humans. It defines a set of processes and associated terminology from an engineering viewpoint. These processes can be applied at any level in the hierarchy of a system's structure. Selected sets of these processes can be applied throughout the life cycle for managing and performing the stages of a system's life cycle. This is accomplished through the involvement of all stakeholders, with the ultimate goal of achieving customer satisfaction.

ISO/IEC/IEEE 15288:2015 also provides processes that support the definition, control and improvement of the system life cycle processes used within an organization or a project. Organizations and projects can use these processes when acquiring and supplying systems

ISO/IEC/IEEE 15288:2015 concerns those systems that are man-made and may be configured with one or more of the following system elements: hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials and naturally occurring entities.

ISO/IEC 9834-8:2014 Information technology -- Procedures for the operation of object identifier registration authorities -- Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers

return to the ISO Standards

Table 63: Data sheet for Information technology – Procedures for the operation of object identifier registration authorities – Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers

Title	Information technology – Procedures for the operation of object identifier registration authorities – Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers
Acronym	
Version	2014
Document Number	9834-8:2014
Release Date	2014-08
Reference	https://www.iso.org/standard/35736.html

Note: The following is an excerpt from the official ISO catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

SO/IEC 9834-8:2014 specifies procedures for the generation of universally unique identifiers (UUIDs) and for their use in the international object identifier tree under the joint UUID arc.

International Telecommunications Union (ITU)

return to the Technical Standards area

Create a New Page for ITU **Number** (i.e., Y.2060 →		Add page

Founded in 1865 to facilitate international connectivity in communications networks, we allocate global radio spectrum and satellite orbits, develop the technical standards that ensure networks and technologies seamlessly interconnect, and strive to improve access to ICTs to underserved communities worldwide. Every time you make a phone call via the mobile, access the Internet or send an email, you are benefiting from the work of ITU.

ITU is committed to connecting all the world's people – wherever they live and whatever their means. [169)]

Technical Standards

- ITU-T Y.2060 - Overview of the Internet of things

169)
“About International Telecommunication Union (ITU)”, 27 Jun 2019, https://www.itu.int/en/about/Pages/default.aspx

ITU-T Y.2060 - Overview of the Internet of things

return to the ITU Standards

Table 64: Data sheet for ITU-T Y.2060 Overview of the Internet of things (AAAA)

Title	Overview of the Internet of things
Acronym	
Version	1.0
Document Number	ITU-T Y.2060
Release Date	15 June 2012
Reference	https://www.itu.int/rec/T-REC-Y.2060-201206-I

Note: The following is an excerpt from the official ITU standard or Document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Summary

Recommendation ITU-T Y.2060 provides an overview of the Internet of things (IoT). It clarifies the concept and scope of the IoT, identifies the fundamental characteristics and high-level requirements of the IoT and describes the IoT reference model. The ecosystem and business models are also provided in an informative appendix.

Scope

This Recommendation provides an overview of the Internet of things (IoT) with the main objective of highlighting this important area for future standardization.

More specifically, this Recommendation covers the following:

- *IoT-related terms and definitions*
- *concept and scope of the IoT*
- *characteristics of the IoT*
- *high-level requirements of the IoT*
- *IoT reference models.*

IoT ecosystem and business models-related information is provided in Appendix I. [170)]

[170)]

“Overview of the Internet of things ”, International Telecommunication Union, SERIES Y: GLOBAL

INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS Next Generation Networks – Frameworks and functional architecture models, 15 June 2012, https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items

National Institute of Standards and Technology (NIST)

return to the Technical Standards area

Create a New Page for NIST (e.g. *SP 800-89*) **Number →** Add page

The National Institute of Standards and Technology (NIST) was founded in 1901 and is now part of the U.S. Department of Commerce. NIST is one of the nation's oldest physical science laboratories. Congress established the agency to remove a major challenge to U.S. industrial competitiveness at the time—a second-rate measurement infrastructure that lagged behind the capabilities of the United Kingdom, Germany, and other economic rivals.

From the smart electric power grid and electronic health records to atomic clocks, advanced nanomaterials, and computer chips, innumerable products and services rely in some way on technology, measurement, and standards provided by the National Institute of Standards and Technology.

Today, NIST measurements support the smallest of technologies to the largest and most complex of human-made creations—from nanoscale devices so tiny that tens of thousands can fit on the end of a single human hair up to earthquake-resistant skyscrapers and global communication networks.
https://www.nist.gov/about-nist

Technical Standards

- NIST: FIPS PUB 186-4: Digital Signature Standard (DSS)
- NIST: SP 800-89: Recommendation for Obtaining Assurances for Digital Signature Applications
- NIST: SP 800-126: The Technical Specification for the Security Content Automation Protocol (SCAP)

NIST: FIPS PUB 186-4: Digital Signature Standard (DSS)

return to the NIST Standards

Table 65: Data sheet for FIPS PUB 186-4: Digital Signature Standard (DSS)

Title	Digital Signature Standard
Acronym	DSS
Version	2013
Series	FIPS
Document Number	FIPS PUB 186-4
Release Date	July 2013
Download	https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

Note: The following is an excerpt from the official NIST catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

This Standard defines methods for digital signature generation that can be used for the protection of binary data (commonly called a message), and for the verification and validation of those digital signatures. Three techniques are approved.

1. *The Digital Signature Algorithm (DSA) is specified in this Standard. The specification includes criteria for the generation of domain parameters, for the generation of public and private key pairs, and for the generation and verification of digital signatures.*
2. *The RSA digital signature algorithm is specified in American National Standard (ANS) X9.31 and Public Key Cryptography Standard (PKCS) #1. FIPS 186-4 approves the use of implementations of either or both of these standards and specifies additional requirements.*
3. *The Elliptic Curve Digital Signature Algorithm (ECDSA) is specified in ANS X9.62. FIPS 186-4 approves the use of ECDSA and specifies additional requirements. Recommended elliptic curves for Federal Government use are provided herein.*

This Standard includes requirements for obtaining the assurances necessary for valid digital signatures. Methods for obtaining these assurances are provided in NIST Special Publication (SP) 800-89, Recommendation for Obtaining Assurances for Digital Signature Applications.

NIST: SP 800-89: Recommendation for Obtaining Assurances for Digital Signature Applications

return to the NIST Standards

Table 66: Data sheet for Recommendation for Obtaining Assurances for Digital Signature Applications

Title	Recommendation for Obtaining Assurances for Digital Signature Applications
Acronym	
Version	
Series	SP
Document Number	800-89
Release Date	November 2006
Download	https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-89.pdf

Note: The following is an excerpt from the official NIST catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

A digital signature is an electronic analogue of a written signature; the digital signature can be used to provide assurance that the claimed signatory signed the information. In addition, a digital signature may be used to detect whether or not the information was modified after it was signed (i.e., to detect the integrity of the signed data). Each signatory has a public and private key and is the owner of that key pair. The private key is used by the owner to generate a digital signature; the public key is used in the signature verification process.

Entities participating in the generation or verification of digital signatures depend on the authenticity of the process. This Recommendation specifies methods for obtaining the assurances necessary for valid digital signatures: assurance of domain parameter validity, assurance of public key validity, assurance that the key pair owner actually possesses the private key, and assurance of the identity of the key pair owner.

Federal Information Processing Standard (FIPS) 186-3 allows three techniques for the generation of digital signatures: the Digital Signature Algorithm (DSA), RSA [171)], and the Elliptic Curve Digital Signature Algorithm (ECDSA). For DSA and ECDSA, assurance of the validity of the domain parameters must be obtained (see Section 4). RSA has no domain parameters. Domain parameters may be generated by anyone, and assurance of domain parameter validity `shall` *be obtained prior to performing any other process associated with digital signatures, including the generation of digital signature key pairs and the generation and verification of digital signatures.*

Digital signature keys may be generated by the intended signatory, cooperatively generated by the intended signatory and a Trusted Third Party (TTP), or generated entirely by a TTP and provided to

the intended signatory. For all digital signature algorithms, each party associated with the digital signature process `shall` have assurance of the validity of the public keys (see Section 5) and assurance that the key pair owner actually possesses the private key used for generating the digital signature (see Section 6). In addition, assurance of the claimed signatory's identity is required by all verifiers, including any TTPs involved in the process (see Section 7).

All methods that are used to provide assurance assume the security and reliability of any routines involved in the process. Obtaining assurances normally requires explicit actions by someone. However, once the appropriate assurances are obtained, the explicitly obtained assurance can be leveraged as assurance for subsequent messages. Note that it may be appropriate to renew this assurance periodically.

171)

An algorithm developed by Rivest, Shamir and Adelman.

NIST: SP 800-126: The Technical Specification for the Security Content Automation Protocol (SCAP)

return to the NIST Standards

Table 67: Data sheet for The Technical Specification for the Security Content Automation Protocol (SCAP)

Title	The Technical Specification for the Security Content Automation Protocol
Acronym	SCAP
Version	1.3
Series	SP
Document Number	800-126
Release Date	2 February 2019
Overview Page	
Download	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-126r3.pdf

Note: The following is an excerpt from the official NIST catalog. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose and Scope

This document, NIST Special Publication (SP) 800-126 Revision 3, along with its annex (NIST SP 800- 126A [SP800-126A]) and a set of schemas, collectively provide the definitive technical specification for version 1.3 of the Security Content Automation Protocol (SCAP). SCAP (pronounced ess-cap) consists of a suite of specifications for standardizing the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans. This document defines requirements for creating and processing SCAP source content. These requirements build on the requirements defined within the individual SCAP component specifications. Each new requirement pertains either to using multiple component specifications together or to further constraining one of the individual component specifications. The requirements within the individual component specifications are not repeated in this document; see those specifications to view their requirements.

To extend the contents of this document, an annex has been created. The annex document specifies additional entities that may be used in SCAP 1.3 conformant content creation and processing:

- *Particular minor version updates to SCAP 1.3 component specifications*
- *Particular Open Vulnerability and Assessment Language (OVAL) platform schema versions*

The scope of this document and its annex is limited to SCAP version 1.3. Other versions of SCAP and its component specifications are not addressed in these documents. Future versions of SCAP will be defined in distinct revisions of this document and its annex, each clearly labeled with a

document revision number and the appropriate SCAP version number.

Organization for the Advancement of Structured Information Standards (OASIS)

return to the Technical Standards area

Create a New Page for OASIS (e.g. *SAML*) **Acronym** → Add page

OASIS is a nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society.

OASIS promotes industry consensus and produces worldwide standards for security, Internet of Things, cloud computing, energy, content technologies, emergency management, and other areas. OASIS open standards offer the potential to lower cost, stimulate innovation, grow global markets, and protect the right of free choice of technology.

OASIS members broadly represent the marketplace of public and private sector technology leaders, users and influencers. The consortium has more than 5,000 participants representing over 600 organizations and individual members in more than 65 countries.

OASIS is distinguished by its transparent governance and operating procedures. Members themselves set the OASIS technical agenda, using a lightweight process expressly designed to promote industry consensus and unite disparate efforts. Completed work is ratified by open ballot. Governance is accountable and unrestricted. Officers of both the OASIS Board of Directors and Technical Advisory Board are chosen by democratic election to serve two-year terms. Consortium leadership is based on individual merit and is not tied to financial contribution, corporate standing, or special appointment.https://www.oasis-open.org/org

Technical Standards

- OASIS: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)
- OASIS: eXtensible Access Control Markup Language (XACML)

OASIS: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)

return to the OASIS Standards

Table 68: Data sheet for Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)

Title	Assertions and Protocols for the OASIS Security Assertion Markup Language
Acronym	SAML
Version	2.0
OASIS Document Number	saml-core-2.0-os
Release Date	15 March 2005
About Specification	https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
Document	https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This specification defines both the structure of SAML assertions, and an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML [XML] and use XML namespaces [XMLNS]. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. The SAML bindings specification [SAMLBind] provides frameworks for the embedding and transport of SAML protocol messages. The SAML profiles specification [SAMLProf] provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

For additional explanation of SAML terms and concepts, refer to the SAML technical overview [SAMLTechOvw] and the SAML glossary [SAMLGloss] . Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMLP-XSD] are also available. The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

OASIS: eXtensible Access Control Markup Language (XACML)

return to the OASIS Standards

Table 69: Data sheet for eXtensible Access Control Markup Language (XACML)

Title	eXtensible Access Control Markup Language (XACML)
Acronym	XACML
Version	3.0
OASIS Document Number	acml-3.0-core-spec-os
Release Date	22 January 2013
About Specification	https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
Document	http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The eXtensible Access Control Markup Language (XACML) [XACML-3] defines an architecture and a language for access control (authorization). The language consists of requests, responses, and policies. Clients send a request to a server to query whether a given action should be allowed. The server evaluates the request against the available policies and returns a response. The policies implement the organization's access control requirements.https://tools.ietf.org/html/rfc7061

Object Management Group (OMG)

return to the Technical Standards area

Create a New Page for OMG (e.g. *DDS*) **Acronym** →		Add page

The Object Management Group® (OMG®) is an international, open membership, not-for-profit technology standards consortium.

Founded in 1989, OMG standards are driven by vendors, end-users, academic institutions and government agencies. OMG Task Forces develop enterprise integration standards for a wide range of technologies and an even wider range of industries.

Technical Standards

- OMG: Automated Source Code CISQ Measures (ASCQM)
- OMG: Automated Source Code CISQ Maintainability Measure (ASCMM)
- OMG: Automated Source Code CISQ Security Measure (ASCSM)
- OMG: Automated Source Code CISQ Performance Efficiency Measure (ASCPEM)
- OMG: Automated Source Code CISQ Reliability Measure (ASCRM)
- OMG: CISQ Automated Enhancement Points (AEP)
- OMG: CISQ Automated Function Points (AFP)
- OMG: CISQ Automated Technical Debt Measure (ATDM)
- OMG: Case Management Model and Notation (CMMN)
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)
- OMG: Interface Definition Language (IDL)
- OMG: Ontology Definition Metamodel (ODM)
- OMG: Semantics Of Business Vocabulary and Rules (SBVR)
- OMG: Structured Assurance Case Metamodel (SACM)
- OMG: Structured Metrics Metamodel (SMM)
- OMG: Systems Modeling Language (SysML)
- OMG: Unified Architecture Framework (UAF)

OMG: Automated Source Code CISQ Measures (ASCQM)

return to the OMG Standards

Table 70: Automated Source Code Measures (ASCQM)

Title	Automated Source Code Quality Measures
Acronym	ASCQM
Version	1.0
OMG Document Number	formal/20-01-02
Release Date	January 2020
About Specification	https://www.omg.org/spec/ASCQM/About-ASCQM
Document	https://www.omg.org/spec/ASCQM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope/Purpose

This specification updates, expands, and combines four previous adopted specifications of the OMG:

- *Automated Source Code Maintainability Measure (ASCMM)* https://www.omg.org/spec/ASCMM/1.0/
- *Automated Source Code Performance Efficiency Measure (ASCPEM)* https://www.omg.org/spec/ASCPEM/1.0/
- *Automated Source Code Reliability Measure (ASCRM)* https://www.omg.org/spec/ASCRM/1.0/
- *Automated Source Code Security Measure (ASCSM)*https://www.omg.org/spec/ASCSM/1.0/

The measures in these standards were calculated from detecting and counting violations of good architectural and coding practices in the source code that could result in unacceptable operational risks or excessive costs. Establishing standards for these measures at the source code level is important because they have been used in outsourcing and system development contracts without having international standards to reference. For instance, the ISO/IEC 25000 series of standards that govern software product quality do not provide measures at the source code level.

A primary objective of updating these measures was to extend their applicability to embedded software, which is especially important for the growing implementation of embedded devices and the Internet of Things. Functionality that has traditionally been implemented in IT applications is now being moved to embedded chips. Since the weaknesses included in the measures specified in this document have been found to be applicable to all forms of software, embedded software is not

treated separately in this specification.

OMG: Automated Source Code CISQ Maintainability Measure (ASCMM)

return to the OMG Standards

Table 71: Data sheet for Automated Source Code CISQ Maintainability Measure (ASCMM)

Title	Automated Source Code CISQ Maintainability Measure
Acronym	ASCMM
Version	1.0
OMG Document Number	formal/2016-01-01
Release Date	December 2016
About Specification	https://www.omg.org/spec/ASCMM/
Document	https://www.omg.org/spec/ASCMM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The purpose of this specification is to establish a standard measure of Maintainability based on detecting violations of good architectural and coding practices that could result in unreliable operation such as outages, data corruption, and lengthy recovery from system failures. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as The Internet of Things. The Consortium for IT Software Quality (CISQ) was formed as a special interest group of OMG to create specifications for automating standard measures of software quality attributes and submit them to OMG for approval.

OMG: Automated Source Code CISQ Security Measure (ASCSM)

return to the OMG Standards

Table 72: Data sheet for Automated Source Code CISQ Security Measure (ASCSM)

Title	Automated Source Code CISQ Security Measure
Acronym	ASCSM
Version	1.0
OMG Document Number	formal/16-01-04
Release Date	December 2016
About Specification	https://www.omg.org/spec/ASCSM/
Document	https://www.omg.org/spec/ASCSM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The purpose of this specification is to establish a standard measure of security based on detecting violations of good architectural and coding practices that could result in unauthorized entry into systems, theft of confidential information, and the malicious compromise of system integrity. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as The Internet of Thing Consortium.

OMG: Automated Source Code CISQ Performance Efficiency Measure (ASCPEM)

return to the OMG Standards

Table 73: Data sheet for Automated Source Code CISQ Performance Efficiency Measure (ASCPEM)

Title	Automated Source Code CISQ Performance Efficiency Measure
Acronym	ASCPEM
Version	1.0
OMG Document Number	formal/16-01-02
Release Date	December 2016
About Specification	https://www.omg.org/spec/ASCPEM/
Document	https://www.omg.org/spec/ASCPEM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose

The purpose of this specification is to establish a standard measure of Performance Efficiency based on detecting violations of good architectural and coding practices that could result in inefficient operation such as performance degradation or excessive use of processor resources. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as The Internet of Things. The Consortium for IT Software Quality (CISQ) was formed as a special interest group of OMG to create specifications for automating standard measures of software quality attributes and submit them to OMG for approval.

OMG: Automated Source Code CISQ Reliability Measure (ASCRM)

return to the OMG Standards

Table 74: Data sheet for Automated Source Code CISQ Reliability Measure (ASCRM)

Title	Automated Source Code CISQ Reliability Measure
Acronym	ASCRM
Version	1.0
OMG Document Number	formal/16-01-03
Release Date	December 2016
About Specification	https://www.omg.org/spec/ASCRM/
Document	https://www.omg.org/spec/ASCRM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The purpose of this specification is to establish a standard measure of reliability based on detecting violations of good architectural and coding practices that could result in unreliable operation such as outages, data corruption, and lengthy recovery from system failures. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as The Internet of Things. The Consortium for IT Software Quality (CISQ) was formed as a special interest group of OMG to create specifications for automating standard measures of software quality attributes and submit them to OMG for approval.

OMG: CISQ Automated Enhancement Points (AEP)

return to the OMG Standards

Table 75: Data sheet for AEP

Title	Automated Enhancement Points
Acronym	AEP
Version	1.0
OMG Document Number	formal/17-04-03
Release Date	April 2017
About Specification	https://www.omg.org/spec/AEP/
Document	https://www.omg.org/spec/AEP/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose

The purpose of this specification is to establish an automated sizing measure, Automated Enhancement Points, which solves specific problems with OMG's approved specification for Automated Function Points when used to measure maintenance and enhancement work performed between two revisions of the software. Current functional sizing measures frequently produce inaccurate and misleading results when looking at functional size evolution and comparing the value to the effort expended during maintenance and enhancement. This problem is especially acute for analyzing productivity or evaluating contract performance. The solution proposed in Automated Enhancement Points establishes a standard measure that addresses 1) the complexity of the requested change, 2) anomalies in counting practices, and 3) measurement of the non-functional elements of an application. Automated Enhancement Points extend beyond the user transactional functions of a business application, the domain of functional measurement, to incorporate the components and elements that manage the non-functional, structural requirements of the application and allow it to perform in a modern multi-technology, multi-platform environment. As an additional benefit, this expansion beyond functional elements provides a foundation for extending automated software size measurement to organizations such as the Industrial Internet Consortium that involve highly algorithmic or embedded software, a domain that traditionally measured size only in lines of code.

OMG: CISQ Automated Function Points (AFP)

return to the OMG Standards

Table 76: Data sheet for Automated Function Points (AFP)

Title	Automated Function Points
Acronym	AFP
Version	1.0
OMG Document Number	formal/2014-01-03
Release Date	December 2014
About Specification	https://www.omg.org/spec/AFP
Document	https://www.omg.org/spec/AFP/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose

This specification defines a method for automating the counting of Function Points that is generally consistent with the Function Point Counting Practices Manual, Release 4.3.1 (IFPUG CPM) produced by the International Function Point Users Group (IFPUG). Guidelines in this specification may differ from those in the IFPUG CPM at points where subjective judgments have to be replaced by the rules needed for automation. The IFPUG CPM was selected as the anchor for this specification because it is the most widely used functional measurement specification with a large supporting infrastructure maintained by a professional organization.

Has also been published as an ISO standard: ISO/IEC 19515:2019(E). Was the first specification submitted to OMG by CISQ.

OMG: CISQ Automated Technical Debt Measure (ATDM)

return to the OMG Standards

Table 77: Data sheet for Automated Technical Debt Measure (ATDM)

Title	Automated Technical Debt Measure
Acronym	ATDM
Version	1.0
OMG Document Number	formal/18-09-01
Release Date	December 2017
About Specification	https://www.omg.org/spec/ATDM/About-ATDM/
Document	https://www.omg.org/spec/ATDM/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose

The purpose of this specification is to establish a standard for automating a measure of Technical Debt that can be computed by source code analysis technologies which have implemented the CISQ Quality Characteristic measures. Within this defined focus, Technical Debt is calculated as an estimate of the effort to fix violations of good architectural and coding practices that must be remediated because of their risk and cost to the business. The foundation for specifying this measure has been provided in the CISQ Quality Characteristic measures approved as OMG standards, namely the Automated Source Code Reliability/Security/Performance Efficiency/Maintainability Measures. Using these OMG standards to provide the content for a measure of Technical Debt allows it to be based on published standards.
Adoption of the Technical Debt metaphor is growing as a means of communicating between IT executives and their technical staffs about quality issues and costs. Commercial IT executives have embraced the concept of Technical Debt for its value in predicting such factors as the costs of future corrective maintenance and the difficulty of enhancing or scaling an application. Currently, several static analysis vendors have added a measure of Technical Debt to their features, but none of these measures are based on an approved international standard.

OMG: Case Management Model and Notation (CMMN)

return to the OMG Standards

Table 78: Data sheet for Case Management Model and Notation (CMMN)

Title	Case Management Model and Notation (CMMN)
Acronym	CMMN
Version	1.1
OMG Document Number	formal/16-12-01
Release Date	December 2016
About Specification	https://www.omg.org/spec/CMMN
Document	https://www.omg.org/spec/CMMN/1.1/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines a common meta-model and notation for modeling and graphically expressing a Case, as well as an interchange format for exchanging Case models among different tools. The specification is intended to capture the common elements that Case management products use, while also taking into account current research contributions on Case management. It is to Case management products what the OMG Business Process Model and Notation (BPMN) specification is to business process management products. This specification is intended to be consistent with and complementary to BPMN.

BPMN has been adopted as a business process modeling standard. It addresses capabilities incorporated in a number of other business process modeling languages, where processes are described as the predefined sequences of activities with decisions (gateways) to direct the sequence along alternative paths or for iterations. These models are effective for predefined, fully specified, repeatable business processes.

For some time, there has been discussion of the need to model activities that are not so predefined and repeatable, but instead depend on evolving circumstances and ad hoc decisions by knowledge workers regarding a particular situation, a case (see Davenport 1994 and 2005; and Van der Aalst 2005). Applications of Case management include licensing and permitting in government, application and claim processing in insurance, patient care and medical diagnosis in healthcare, mortgage processing in banking, problem resolution in call centers, sales and operations planning, invoice discrepancy handling, maintenance and repair of machines and equipment, and engineering of made-to-order products.

OMG: Data Distribution Service (DDS)

return to the OMG Standards

Table 79: Data sheet for Data Distribution Service (DDS)

Title	Data Distribution Service
Acronym	DDS
Version	1.4
OMG Document Number	formal/15-04-10
Release Date	March 2015
About Specification	https://www.omg.org/spec/DDS/
Document	https://www.omg.org/spec/DDS/1.4/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The DDS specification describes a Data-Centric Publish-Subscribe (DCPS) model for distributed application communication and integration. This specification defines both the Application Interfaces (APIs) and the Communication Semantics (behavior and quality of service) that enable the efficient delivery of information from information producers to matching consumers.

- *The purpose of the DDS specification can be summarized as enabling the "Efficient and Robust Delivery of the Right Information to the Right Place at the Right Time."*

The expected application domains require DCPS to be high-performance and predictable as well as efficient in its use of resources. To meet these requirements it is important that the interfaces are designed in such a way that they:

- *Allow the middleware to pre-allocate resources so that dynamic resource allocation can be reduced to the minimum,*
- *Avoid properties that may require the use of unbounded or hard-to-predict resources, and*
- *Minimize the need to make copies of the data.*

DDS uses typed interfaces (i.e., interfaces that take into account the actual data types) to the extent possible. Typed interfaces offer the following advantages:

- *They are simpler to use: the programmer directly manipulates constructs that naturally represent the data.*
- *They are safer to use: verifications can be performed at compile time.*
- *They can be more efficient: the execution code can rely on the knowledge of the exact data type it has in advance, to e.g., pre-allocate resources.*

It should be noted that the decision to use typed interfaces implies the need for a generation tool to translate type descriptions into appropriate interfaces and implementations that fill the gap between the typed interfaces and the generic middleware.

QoS (Quality of Service) is a general concept that is used to specify the behavior of a service. Programming service behavior by means of QoS settings offers the advantage that the application developer only indicates 'what' is wanted rather than 'how' this QoS should be achieved. Generally speaking, QoS is comprised of several QoS policies. Each QoS policy is then an independent description that associates a name with a value. Describing QoS by means of a list of independent QoS policies gives rise to more flexibility.

This specification is designed to allow a clear separation between the publish and the subscribe sides, so that an application process that only participates as a publisher can embed just what strictly relates to publication. Similarly, an application process that participates only as a subscriber can embed only what strictly relates to subscription.

OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)

return to the OMG Standards

Table 80: Data sheet for DDS Interoperability Wire Protocol (DDSI-RTPS)

Title	DDS Interoperability Wire Protocol
Acronym	DDSI-RTPS
Version	2.3
OMG Document Number	formal/19-04-03
Release Date	May 2019
About Specification	https://www.omg.org/spec/DDSI-RTPS/
Document	https://www.omg.org/spec/DDSI-RTPS/2.3/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

This specification defines an interoperability wire protocol for DDS. Its purpose and scope is to ensure that applications based on different vendors' implementations of DDS can interoperate.

OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)

return to the OMG Standards

Table 81: Data sheet for ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)

Title	ISO/IEC C++ 2003 Language DDS PSM
Acronym	DDS-PSM-Cxx
Version	1.0
OMG Document Number	formal/13-11-01
Release Date	November 2013
About Specification	https://www.omg.org/spec/DDS-PSM-Cxx/
Document	https://www.omg.org/spec/DDS-PSM-Cxx/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The "ISO/IEC C++ Language DDS Platform Specific Model (PSM)" (DDS-PSM-Cxx) was motivated by mainly two reasons. First the IDL-derived C++ API for DDS does not integrate well with the C++ language and it does not leverage some of the features provided by the C++ language today universally sup- ported by C++ compilers. Second, the current IDL-derived PSM suffers from the gap existing between the features available in IDL and those available in a programming language such as C++. Some examples of this gap are as simple as method overloading, yet, there are many other examples that we could make in comparing the expressiveness power of IDL versus that of native C++.

As a result this specification takes a completely fresh look at how a native C++ PSM can be derived from the DDS Platform Independent Model (PIM). In doing so, it tries to balance two forces - derive an API that is as simple and safe as possible while retaining the structure of the PIM. This specification does not require C++11 features for its implementation, yet it is designed to enable the use of C++11 features, such as the auto keyword, range-based for loops, etc.

OMG: Java 5 Language PSM for DDS (DDS-Java)

return to the OMG Standards

Table 82: Data sheet for Java 5 Language PSM for DDSM (DDS-Java)

Title	Java 5 Language PSM for DDS
Acronym	DDS-Java
Version	1.0
OMG Document Number	formal/13-11-02
Release Date	November 2013
About Specification	https://www.omg.org/spec/DDS-Java/
Document	https://www.omg.org/spec/DDS-Java/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines a Platform Specific Model (PSM) for the OMG Data Distribution Service for Real-Time Systems (DDS). It specifies an API only for the Data-Centric Publish-Subscribe (DCPS) portion of that specification; it does not address the Data Local Reconstruction Layer (DLRL). In addition, it encompasses (a) the DDS APIs introduced by [DDSXTypes] and (b) an API to specifying QoS libraries and profiles such as were specified by [DDS-CCM]. This specification also defines a means of publishing and subscribing Java objects with DDS-the Java Type Representation without first describing the types of those objects in another language, such as XML or OMG IDL.

OMG: OPC-UA/DDS Gateway (DDS-OPCUA)

return to the OMG Standards

Table 83: Data sheet for OPC-UA/DDS Gateway (DDS-OPCUA)

Title	OPC-UA/DDS Gateway
Acronym	DDS-OPCUA
Version	1.0
OMG Document Number	formal/20-01-01
Release Date	February 2020
About Specification	https://www.omg.org/spec/DDS-OPCUA/
Document	https://www.omg.org/spec/DDS-OPCUA/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

Data Distribution Service (DDS) is a family of standards from the Object Management Group (OMG) that provide connectivity, interoperability, and portability for Industrial Internet, cyber-physical, and mission-critical applications.

The DDS connectivity standards cover Publish-Subscribe (DDS), Service Invocation (DDS-RPC), Interoperability (DDS-RTPS), Information Modeling (DDS-XTYPES), Security (DDS-SECURITY), as well as programing APIs for C, C++, Java and other languages.

The OPC Unified Architecture (OPC UA) is an information exchange standard for Industrial Automation and related systems created by the OPC Foundation. The OPC UA standard provides an Addressing and Information Model for Data Access, Alarms, and Service invocation layered over multiple transport-level protocols such as Binary TCP and Web-Services.

DDS and OPC UA exhibit significant deployment similarities:

- *Both enable independently developed applications to interoperate even when those applications come from different vendors, use different programming languages, or run on different platforms and operating systems.*
- *Both have significant traction within Industrial Automation systems.*
- *Both define standard protocols built on top of the TCP/ UDP/IP Internet stacks. The two technologies may coexist within the same application domains; however, while there are solutions that bridge between DDS and OPC UA, these are based on custom mappings and cannot be relied to work across vendors and products.*

This specification overcomes this situation by defining a standard, vendor-independent, configurable gateway that enables interoperability and information exchange between systems that use DDS and systems that use OPC UA.

OMG: RPC Over DDS (DDS-RPC)

return to the OMG Standards

Table 84: Data sheet for RPC Over DDS (DDS-RPC)

Title	RPC Over DDS
Acronym	DDS-RPC
Version	1.0
OMG Document Number	formal/17-04-01
Release Date	April 2017
About Specification	https://www.omg.org/spec/DDS-RPC/
Document	https://www.omg.org/spec/DDS-RPC/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

The Data Distribution Service is widely used for data-centric publish/subscribe communication in real-time distributed systems. Large distributed systems often need more than one style of communication. For instance, data distribution works great for one-to-many dissemination of information. However, certain other styles of communication namely request/reply and remote method invocation are cumbersome to express using the basic building blocks of DDS. Using two or more middleware frameworks is often not practical due to complexity, cost, and maintenance overhead reasons. As a consequence, developing a standard mechanism for request/reply style bidirectional communication on top of DDS is highly desirable for portability and interoperability. Such facility would allow commands to be naturally represented as remote method invocations. This presents a solution to this problem.

This specification defines a RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC) framework using the basic building blocks of DDS, such as topics, types, DataReaders, and DataWriters to provide request/reply semantics. It defines distributed services, described using a service interface, which serves as a shareable contract between service provider and a service consumer. It supports synchronous and asynchronous method invocation. Despite its similarity, it is not intended to be a replacement for CORBA.

OMG: DDS Security (DDS-SECURITY)

return to the OMG Standards

Table 85: Data sheet for DDS Security (DDS-SECURITY)

Title	DDS Security
Acronym	DDS-SECURITY
Version	1.1
OMG Document Number	formal/18-04-01
Release Date	July 2018
About Specification	https://www.omg.org/spec/DDS-SECURITY/1.1/
Document	https://www.omg.org/spec/DDS-SECURITY/1.1/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview of this Specification

This specification defines the Security Model and Service Plugin Interface (SPI) architecture for compliant DDS implementations. The DDS Security Model is enforced by the invocation of these SPIs by the DDS implementation. This specification also defines a set of builtin implementations of these SPIs.

- *The specified builtin SPI implementations enable out-of-the box security and interoperability between compliant DDS applications.*
- *The use of SPIs allows DDS users to customize the behavior and technologies that the DDS implementations use for Information Assurance, specifically customization of Authentication, Access Control, Encryption, Message Authentication, Digital Signing, Logging and Data Tagging.*

This specification defines five SPIs that when combined together provide Information Assurance to DDS systems:

- ***Authentication** Service Plugin. Provides the means to verify the identity of the application and/or user that invokes operations on DDS. Includes facilities to perform mutual authentication between participants and establish a shared secret.*
- ***AccessControl** Service Plugin. Provides the means to enforce policy decisions on what DDS related operations an authenticated user can perform. For example, which domains it can join, which Topics it can publish or subscribe to, etc.*
- ***Cryptographic** Service Plugin. Implements (or interfaces with libraries that implement) all cryptographic operations including encryption, decryption, hashing, digital signatures, etc. This includes the means to derive keys from a shared secret.*
- ***Logging** Service Plugin. Supports auditing of all DDS security-relevant events.*

- ***Data Tagging*** *Service Plugin. Provides a way to add tags to data samples.*

OMG: Web-Enabled DDS (DDS-WEB)

return to the OMG Standards

Table 86: Data sheet for Web-Enabled DDS (DDS-WEB)

Title	Web-Enabled DDS
Acronym	DDS-WEB
Version	1.0
OMG Document Number	formal/16-03-01
Release Date	February 2016
About Specification	https://www.omg.org/spec/DDS-WEB/
Document	https://www.omg.org/spec/DDS-WEB/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Purpose

The goal of this specification is to define the means for applications using standard web protocols to participate as first-class citizens as publishers and subscribers of data in the DDS Global Data space. This participation is realized by exposing a WebDDS Object Model and making it accessible as a web service, a REST resources, or some other standard web protocol. Exposing access via these web-friendly protocols allow applications built on various technology stacks (e.g. JavaScript, Python, PHP, Perl , etc.) to communicate with native DDS applications.

OMG: DDS Consolidated XML Syntax (DDS-XML)

return to the OMG Standards

Table 87: Data sheet for DDS Consolidated XML Syntax (DDS-XML)

Title	DDS Consolidated XML Syntax
Acronym	DDS-XML
Version	1.0
OMG Document Number	ptc/18-05-36
Release Date	December 2018
About Specification	https://www.omg.org/spec/DDS-XML/
Document	https://www.omg.org/spec/DDS-XML/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

Historically various specifications have defined XML syntax to represent particular subsets of DDS-related resources:

- *The DDS4CCM specification defines XML syntax to represent the DDS QoS Policies of DDS Entities: `DomainParticipantQos`, `TopicQos`, `PublisherQos`, `SubscriberQos`, `DataWriterQos`, and `DataReaderQos`.*
- *The DDS-XTYPES specification defines XML syntax to represent DDS Data Types and Data Samples.*
- *The DDS-WEB specification defines XML syntax to represent DDS Applications, DDS Domains, and DDS entities (i.e., DomainParticipant, Topic, Publisher, Subscriber, DataWriter, and DataReader). This specification consolidates all this XML syntax into a single document. There are no significant syntactic changes in this document relative to referenced specifications.*

OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)

return to the OMG Standards

Table 88: Data sheet for DDS For Extremely Resource Constrained Environments (DDS-XRCE)

Title	DDS For Extremely Resource Constrained Environments
Acronym	DDS-XRCE
Version	1.0
OMG Document Number	formal/20-02-01
Release Date	February 2020
About Specification	https://www.omg.org/spec/DDS-XRCE/
Document	https://www.omg.org/spec/DDS-XRCE/1.0/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines a XRCE Protocol between a resource constrained, low-powered device (client) and an Agent (the server). The XRCE Protocol enables the device to communicate with a DDS network and publish and subscribe to topics in a DDS domain via an intermediate service (the XRCE Agent). The specification's purpose and scope are to ensure that applications based on different vendors' implementations of the XRCE Protocol and XRCE Agent are compatible and interoperable.

OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

return to the OMG Standards

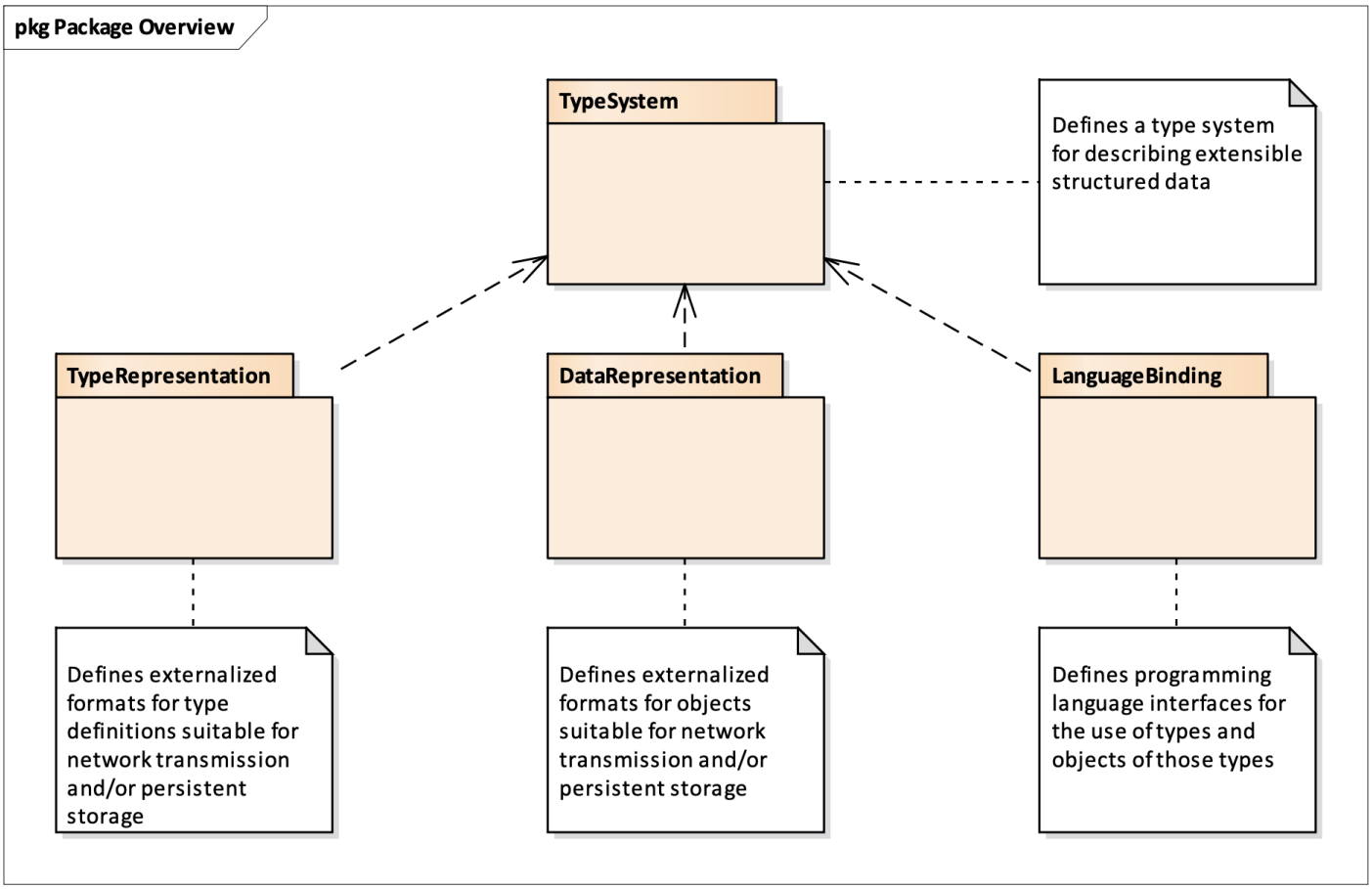
Table 89: Data sheet for Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

Title	Extensible and Dynamic Topic Types for DDS
Acronym	DDS-XTypes
Version	1.3
OMG Document Number	formal/20-02-04
Release Date	February 2020
About Specification	https://www.omg.org/spec/DDS-XTypes/
Document	https://www.omg.org/spec/DDS-XTypes/1.3/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

The Specification addresses four related concerns summarized in the Figure below.



The specification addresses four related concerns: the type system, the representation of types, the representation of data, and the language bindings used to access types and data. Each of these concerns is modeled as a collection of classes belonging to a corresponding package.

This specification provides the following additional facilities to DDS implementations and users:

- **Type System**. The specification defines a model of the data types that can be used for DDS Topics. The type system is formally defined using UML. The Type System is defined in Clause7.2and its sub clauses. The structural model of this system is defined in the Type System Model in Clause7.2.2. The framework under which types can be modified over time is summarized in Clause7.2.3, “Type Extensibility and Mutability.” The concrete rules under which the concepts from 7.2.2and 7.2.3come together to define compatibility in the face of such modifications are defined in Clause7.2.4, “Type Compatibility.”
- **Type Representations**. The specification defines the ways in which types described by the Type System may be externalized such that they can be stored in a file or communicated over a network. The specification adds additional Type Representations beyond the one (IDL [IDL]) already implied by the DDS specification. Several Type Representations are specified in the sub clauses of Clause7.3. These include IDL, XML, XML Schema, and TypeObject.
- **Data Representation**. The specification defines multiple ways in which objects of the types defined by the Type System may be externalized such that they can be stored in a file or communicated over a network. (This is also commonly referred as “data serialization” or “data marshaling.”) The specification extends and generalizes the mechanisms already defined by the

DDS Interoperability specification [RTPS]. The specification includes Data Representations that support data type evolution, that is, allow a data type to change in certain well-defined ways without breaking communication. Two Data Representations are specified in the sub clauses of Clause7.4. These are Extended CDR and XML.

- **Language Binding**. The specification defines multiple ways in which applications can access the state of objects defined by the Type System. The specification extends and generalizes the mechanism currently implied by the DDS specification (“Plain Language Binding”) and adds a Dynamic Language Binding that allows application to access data without compile-time knowledge of its type. The specification also defines an API to define and manipulate data types programmatically. Two Language Bindings are specified in the sub clauses of Clause7.5. These are the Plain Language Binding and the Dynamic Language Binding.

OMG: Interface Definition Language (IDL)

return to the OMG Standards

Table 90: Data sheet for Interface Definition Language (IDL)

Title	Interface Definition Language
Acronym	IDL
Version	4.2
OMG Document Number	formal/18-01-05
Release Date	March 2018
About Specification	https://www.omg.org/spec/IDL/
Document	https://www.omg.org/spec/IDL/4.2/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This document specifies the OMG Interface Definition Language (IDL). IDL is a descriptive language used to define data types and interfaces in a way that is independent of the programming language or operating system/processor platform.

The IDL specifies only the syntax used to define the data types and interfaces. It is normally used in connection with other specifications that further define how these types/interfaces are utilized in specific contexts and platforms:

- *Separate "language mapping" specifications define how the IDL-defined constructs map to specific programming languages, such as, C/C++, Java, C#, etc.*
- *Separate "serialization" specifications define how data objects and method invocations are serialized into a format suitable for network transmission.*
- *Separate "middleware" specifications, such as, DDS or CORBA leverage the IDL to define data-types, services, and interfaces. The description of IDL grammar uses a syntax notation that is similar to Extended Backus-Naur Format (EBNF).*

OMG: Ontology Definition Metamodel (ODM)

return to the OMG Standards

Table 91: Data sheet for Ontology Definition Metamodel

Title	Ontology Definition Metamodel
Acronym	ODM
Version	1.1
OMG Document Number	formal/14-09-02
Release Date	September 2014
About Specification	https://www.omg.org/spec/ODM/
Document	https://www.omg.org/spec/ODM/1.1/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

The authors believe that this specification represents the foundation for an extremely important set of enabling capabilities for Model Driven Architecture (MDA) based software engineering, namely the formal grounding for representation, management, interoperability, and application of business semantics.

The ODM specification offers a number of benefits to potential users, including:

- *Options in the level of expressivity, complexity, and form available for designing and implementing conceptual models, ranging from familiar UML and ER methodologies to formal ontologies represented in description logics or first order logic.*
- *Grounding in formal logic, through standards-based, model-theoretic semantics for the knowledge representation languages supported, sufficient to enable reasoning engines to understand, validate, and apply ontologies developed using the ODM.*
- *Profiles and mappings sufficient to support not only the exchange of models developed independently in various formalisms but to enable consistency checking and validation in ways that have not been feasible to date.*
- *The basis for a family of specifications that marry MDA and Semantic Web technologies to support semantic web services, ontology and policy-based communications and interoperability, and declarative, policy-based applications in general.*

The specification defines a family of independent metamodels, related profiles, and mappings among the metamodels corresponding to several international standards for ontology and Topic Maps definition, as well as capabilities supporting conventional modeling paradigms for capturing conceptual knowledge, such as entity-relationship modeling.
The ODM is applicable to knowledge representation, conceptual modeling, formal taxonomy

development and ontology definition, and enables the use of a variety of enterprise models as starting points for ontology development through mappings to UML and MOF. ODM-based ontologies can be used to support:

- *interchange of knowledge among heterogeneous computer systems,*
- *representation of knowledge in ontologies and knowledge bases,*
- *specification of expressions that are the input to or output from inference engines. The ODM is not intended to encompass*
- *specification of proof theory or inference rules,* * specification of translation and transformations between the notations used by heterogeneous computer systems, * free logics, * conditional logics, * methods of providing relationships between symbols in the logical “universe” and individuals in the “real world,” * issues related to computability using the knowledge representation formalisms represented in the ODM (e.g.,

optimization, efficiency, tractability, etc.).

OMG: Semantics Of Business Vocabulary and Rules (SBVR)

return to the OMG Standards

Table 92: Data sheet for Semantics Of Business Vocabulary and Rules (SBVR)

Title	Semantics Of Business Vocabulary and Rules
Acronym	SBVR
Version	1.5
OMG Document Number	formal/19-10-02
Release Date	December 2019
About Specification	https://www.omg.org/spec/SBVR/
Document	https://www.omg.org/spec/SBVR/1.5/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines the vocabulary and rules (see Clauses 7 through 21) for documenting the semantics of business vocabularies and business rules for the exchange of business vocabularies and business rules among organizations and between software tools.

This specification is interpretable in predicate logic with a small extension using modal operators. It supports linguistic analysis of text for business vocabularies and business rules, with the linguistic analysis itself being outside the scope of this specification.

OMG: Structured Assurance Case Metamodel (SACM)

return to the OMG Standards

Table 93: Data sheet for Structured Assurance Case Metamodel (SACM)

Title	Structured Assurance Case Metamodel
Acronym	SACM
Version	2.1
OMG Document Number	formal/20-04-01
Publication Date	April 2020
About Specification	https://www.omg.org/spec/SACM
Document	https://www.omg.org/spec/SACM/2.1/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines a metamodel for representing structured assurance cases. An Assurance Case is a set of auditable claims, arguments, and evidence created to support the claim that a defined system/service will satisfy the particular requirements. An Assurance Case is a document that facilitates information exchange between various system stakeholder such as suppliers and acquirers, and between the operator and regulator, where the knowledge related to the safety and security of the system is communicated in a clear and defendable way. Each assurance case should communicate the scope of the system, the operational context, the claims, the safety and/or security arguments, along with the corresponding evidence.

Systems Assurance is the process of building clear, comprehensive, and defensible arguments regarding the safety and security properties of systems. The vital element of Systems Assurance is that it makes clear and well-defined claims about the safety and security of systems. Certain claims are supported through reasoning. Reasoning is expressed by explicit annotated links between claims, where one or more claims (called sub-claims) when combined provide inferential support to a larger claim. Certain associations (recorded as assertions) between claims and subclaims can require supporting arguments of their own (e.g., justification of an asserted inference). Claims are propositions which are expressed by statements in some natural language. The degree of precision in formulation of the claims may contribute to the comprehensiveness of an assurance case. The context is important to communicate the scope of the claim, and to clarify the language used by the claim by providing necessary definition and explanations. Context involves assumptions made about the system and its environment. Explicit statement of the assumptions contributes to the comprehensiveness of the argument. Argumentation flow between claims is structured to facilitate communication of the entire assurance case.

OMG: Structured Metrics Metamodel (SMM)

return to the OMG Standards

Table 94: Data sheet for Structured Metrics Metamodel (SMM)

Title	Structured Metrics Metamodel
Acronym	SMM
Version	1.2
OMG Document Number	formal/18-05-01
Release Date	March 2018
About Specification	https://www.omg.org/spec/SMM/
Document	https://www.omg.org/spec/SMM/1.2/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

This specification defines a metamodel for representing measurement information related to any structured information model. Referred to as the Structured Metrics Metamodel (SMM), this specification is an extensible metamodel for exchanging both measures and measurement information concerning artifacts contained or expressed by structured models, such as MOF.

The SMM include elements representing the concepts needed to express a wide range of diversified measures. The specification does include a group of sample measures, but it is not asserting that the listed measures constitute standards themselves; these are supplied simply as non-normative examples. The SMM is a specification for the definition of measures and the representation of their measurement results. A library of measures consists of measure definitions and serves to establish the specification upon which all of the measurements will be based.

The SMM is part of the Architecture Driven Modernization (ADM) roadmap and fulfills the metric needs of the ADM roadmap scenarios as well as other information technology scenarios. SMM's scope, however, is broader than software modeling. This standard looks to fulfill the metric needs across the OMG's wide variety of interest from automotive and business architecture to space and telecommunications.

SMM measures describe methods of computing comparable values such as:

- *Counts (Votes in an election and lines of code measures exemplify the mechanism.)*
- *Direct applications of named measurements (One such named measure is Cyclomatic Complexity.)*
- *Simple algebraic change of calibration of already defined numeric measures. (e.g., the translation to miles from kilometers).*

- *Simple algebraic aggregations of numeric artifact features, including other measures, over sets of artifacts. (Determining an enterprise global sales by summing its regional sales.)*
- *Simple range-based grading or classification of already defined numeric measures. (Exams are frequently measured on a scale of 0 to 100 which is translated to A, B, C, D, and F grades.)*
- *Qualitative evaluations where the range of evaluations can be mapped to a linear order.*

The SMM specifies the representation of measures without detailing the representation of the entities measured. SMM anticipates that those entities are represented in other OMG metamodels. Measured artifacts or their features may be defined within Knowledge Discovery Metamodel (KDM), Abstract Syntax Tree Metamodel (ASTM), Value Delivery Modeling Language (VDML), other OMG metamodels, or other structured models.

The information captured in OMG models often evolves over time. Given the predicate value of metrics with respect to "downstream" problems, metrics are gathered into trends or viewed from historical perspective. As shown in 17.2.1 Historic and Trend Data, SMM addresses the issues of trend and history to model for system development as long as the historical links of the measured entities are provided.

Consistent with other models defined by OMG, the SMM will be defined using the MOF meta-modeling language. As such, it will have a standard textual representation presented by XMI. Consequently, the exchange of metrics defined by SMM will be in the XMI. These models will, similarly, be compatible with MOF repositories for storage and retrieval by various tools.

OMG: Systems Modeling Language (SysML)

return to the OMG Standards

Table 95: Data sheet for Systems Modeling Language (SysML)

Title	OMG Systems Modeling Language (OMG SysML)
Version	1.6
OMG Document Number	formal/19-11-01
Release Date	December 2019
Normative Reference	http://www.omg.org/spec/SysML/1.6/
Machine Consumable File(s)	http://www.omg.org/spec/SysML/20181001

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

The purpose of this International Standard is to specify the Systems Modeling Language (SysML), a general-purpose modeling language for systems engineering. Its intent is to specify the language so that systems engineering modelers may learn to apply and use SysML; modeling tool vendors may implement and support SysML; and both can provide feedback to improve future versions. Note that a definition of “system” and “systems engineering” can be found in ISO/IEC 15288.

SysML reuses a subset of UML 2.5 and provides additional extensions to address the requirements in UML for SE. SysML uses the UML 2.5 extension mechanisms as further elaborated in Clause 17 as the primary mechanism to specify the extensions to UML 2.5. This revision of SysML relies on several new features incorporated into UML 2.5. Any use of the term “UML 2” or “UML” in this specification, unless otherwise noted, will refer to UML 2.5 in general and the UML 2.5 specification in particular.

Since SysML uses UML 2.5 as its foundation, systems engineers modeling with SysML and software engineers modeling with UML 2.5 will be able to collaborate on models of software-intensive systems. This will improve communication among the various stakeholders who participate in the systems development process and promote interoperability among modeling tools. It is anticipated that SysML will be customized to model domain-specific applications, such as automotive, aerospace, communication, and information systems.

SysML is designed to provide simple but powerful constructs for modeling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior, allocations, and constraints on system properties to support engineering analysis. The language is intended to support multiple processes and methods such as structured, object-oriented, and others, but each methodology may impose additional constraints on how a construct or diagram kind may be used. This version of the language supports most, but not all, of the requirements of

the UML for Systems Engineering Request For Proposal (RFP), as shown in the Requirements Traceability referenced by Annex F. These gaps are intended to be addressed in future versions of SysML as indicated in the matrix.

OMG: Unified Architecture Framework (UAF)

return to the OMG Standards

Table 96: Data sheet for Unified Architecture Framework (UAF)

Title	Unified Architecture Framework
Acronym	UAF
Version	1.2 beta
OMG Document Number	ptc/19-04-03
Release Date	October 2019
About Specification	https://www.omg.org/spec/UCM/
Document	https://www.omg.org/spec/UCM/1.2/Beta1/PDF

Note: The following is an excerpt from the actual document. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Scope

The scope of Unified Architecture Framework Profile (UAFP) includes the language extensions to enable the extraction of specified and custom models from an integrated architecture description (AD). The models describe a system [172)] from a set of stakeholders' concerns such as security or information through a set of predefined viewpoints and associated views [173)]. Developed models can also reflect custom viewpoints or to develop more formal extensions for new viewpoints. The UAFP specification supports the Department of Defense Architecture Framework (DoDAF) 2.02, the Ministry of Defence Architecture Framework (MODAF), Security Views from Canada's Department of National Defense Architecture Framework (DNDAF) and the North Atlantic Treaty Organization (NATO) Architecture Framework (NAF) v 3.1. The core concepts in the UAF domain metamodel specify the UAFP based upon the DoDAF 2.0.2 Domain Metamodel (DM2) and the MODAF ontological data exchange mechanism (MODEM). MODEM is intended to provide the basis for the next version of NAF). The intent is to provide a standard representation for AD support for Defense Organizations. The intention of UAFP is also to support a standard representation for non-defense organizations' ADs as part of their Systems Engineering (SE) technical processes. The associated UAF metamodel (see C4i-2016-02-03) intent is to improve the ability to exchange architecture data between related tools that are UML/SysML based and tools that are based on other standards.

UAFP v 1.0 supports the capability to:

- *model architectures for a broad range of complex systems, which may include hardware, software, data, personnel, and facility elements;*
- *model consistent architectures for system-of-systems (SoS) down to lower levels of design and implementation;*
- *support the analysis, specification, design, and verification of complex systems; and*
- *improve the ability to exchange architecture information among related tools that are*

SysML based and tools that are based on other standards.

172)

The term system is used from: “Systems and software engineering – Architecture description,” http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

173)

Stakeholder, concern, viewpoint, view and model are also used from “Systems and software engineering – Architecture description,” http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

Open Source Initiative (OSI)

return to the Technical Standards area

Create a New Page for OSI (e.g. *MIT*) **Acronym →** Add page

About

The Open Source Initiative (OSI) is a California public benefit corporation, with 501©3 tax-exempt status, founded in 1998.

It is involved in Open Source community-building, education, and public advocacy to promote awareness and the importance of non-proprietary software. OSI Board members frequently travel the world to attend Open Source conferences and events, meet with open source developers and users, and to discuss with executives from the public and private sectors about how Open Source technologies, licenses, and models of development can provide economic and strategic advantages.[174)]

Mission

The Open Source Initiative (OSI) is a non-profit corporation with global scope formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community.

Open source enables a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, better reliability, greater flexibility, lower cost, and an end to predatory vendor lock-in.

One of our most important activities is as a standards body, maintaining the Open Source Definition for the good of the community. The Open Source Initiative Approved License trademark and program creates a nexus of trust around which developers, users, corporations and governments can organize open source cooperation.[175)]

Technical Standards

There are currently just under 100 licenses approved by the OSI. However, there is a subset which are popular, widely used, or have strong communities:

- Apache License, Version 2.0 (Apache-2.0)
- OSI: The 2-Clause BSD License (BSD-2-Clause)
- OSI: The 3-Clause BSD License (BSD-3-Clause)

- OSI: GNU Library General Public License version 2 (LGPL-2.0)
- OSI: GNU Lesser General Public License version 2.1 (LGPL-2.1)
- OSI: GNU General Public License version 3 (GPL-3.0)
- OSI: The MIT License (MIT)
- OSI: Common Public License, Version 1.0 (CPL-1.0)
- OSI: Eclipse Public License Version 2.0 (EPL-2.0)
- OSI: Mozilla Public License (MPL-2.0)

[174)] , [175)]

About Open Source Initiative

OSI: The 2-Clause BSD License (BSD-2-Clause)

return to the Open Source Initiative area

Comparison of free and open source licenses[176)]

Table 97: Data Sheet for The 2-Clause BSD License (BSD-2-Clause)

License Attribute	Value
Full Name	The 2-Clause BSD License
SPDX Short Name	BSD-2-Clause
Author	OSI Web Archive BSD 2 Clause
Latest Version	1.0
Publication Date	1994-2009
Author URI	Regents of the University of California]]
OSI URI	https://opensource.org/licenses/BSD-2-Clause
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Manually
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	Manually

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/BSD-2-Clause

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE FREEBSD PROJECT ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY

AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

176)

Wikipedia - Comparison of free and open-source software licenses

OSI: The 3-Clause BSD License (BSD-3-Clause)

return to the Open Source Initiative area

Comparison of free and open source licenses[177)]

Note

This license has also been called the “New BSD License” or “Modified BSD License”. See also the 2-Clause BSD License.

Table 98: Data Sheet for The 3-Clause BSD License (BSD-3-Clause)

License Attribute	Value
Full Name	The 3-Clause BSD License
SPDX Short Name	BSD-3-Clause
Author	OSI Web Archive BSD 3 Clause
Latest Version	3.0
Publication Date	2009
Author URI	Regents of the University of California
OSI URI	https://opensource.org/licenses/BSD-3-Clause
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Manually
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	Manually

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/BSD-3-Clause

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

177)

Wikipedia - Comparison of free and open-source software licenses

OSI: GNU Library General Public License version 2 (LGPL-2.0)

return to the Open Source Initiative area

Comparison of free and open source licenses[178)]

Note:

The Free Software Foundation, license steward of the LGPL, regards this license as having been superseded by OSI: GNU General Public License version 3 (GPL-3.0).

Table 99: Data Sheet for GNU Library General Public License version 2 (LGPL-2.0)

License Attribute	Value
Full Name	GNU Library General Public License version 2
SPDX Short Name	LGPL-2.0
Author	https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html
Latest Version	2.0
Publication Date	June 1991
Author URI	https://www.gnu.org/
OSI URI	https://opensource.org/licenses/LGPL-2.0
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/LGPL-2.0

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software

Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not

effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

*A **"library"** means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.*

*The **"Library"**, below, refers to any such software library or work which has been distributed under these terms. A **"work based on the Library"** means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)*

***"Source code"** for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.*

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library

(independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> *a) The modified work must itself be a software library.*
> *b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.*
> *c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.*
> *d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.*

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a ***"work that uses the Library"*** *with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.*

When a ***"work that uses the Library"*** *uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.*

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is

unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

> *a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)*
> *b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.*
> *c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.*
> *d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.*

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

> *a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.*
> *b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.*

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right

claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY

COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Library General Public License for more details.

You should have received a copy of the GNU Library General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

178)

Wikipedia - Comparison of free and open-source software licenses

OSI: GNU Lesser General Public License version 2.1 (LGPL-2.1)

return to the Open Source Initiative area

Comparison of free and open source licenses[179)]

Table 100: Data Sheet for GNU Lesser General Public License version 2.1 (LGPL-2.1)

License Attribute	Value
Full Name	GNU Lesser General Public License version 2.1
SPDX Short Name	LGPL-2.1
Author	https://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
Latest Version	2.1
Publication Date	February 1999
Author URI	https://www.gnu.org/
OSI URI	https://opensource.org/licenses/LGPL-2.1
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/LGPL-2.1

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it;

that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the “Lesser” General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs

must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> *a) The modified work must itself be a software library.*
> *b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.*
> *c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.*
> *d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.*

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under

terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with

any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if

written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Libraries If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the library's name and an idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser

General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

179)

Wikipedia - Comparison of free and open-source software licenses

OSI: GNU General Public License version 3 (GPL-3.0)

return to the Open Source Initiative area

Comparison of free and open source licenses[180)]

Table 101: Data Sheet for GNU General Public License version 3 (GPL-3.0)

License Attribute	Value
Full Name	GNU General Public License version 3
SPDX Short Name	GPL-3.0-only
Author	https://www.gnu.org/licenses/gpl-3.0.en.html
Latest Version	3.0
Publication Date	2007
Author URI	https://www.fsf.org/
OSI URI	https://opensource.org/licenses/GPL-3.0
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/GPL-3.0

Begin license text.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public

Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” *refers to version 3 of the GNU General Public License.*

"Copyright" *also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.*

"The Program"*refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.*

To ***"modify"*** *a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.*

A ***"covered work"*** *means either the unmodified Program or a work based on the Program.*

To ***"propagate"*** *a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.*

To ***"convey"*** *a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.*

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The ***"source code"*** *for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.*

A ***"Standard Interface"*** *means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.*

The ***"System Libraries"*** *of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.*

*The **“Corresponding Source”** for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.*

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties'

legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> *a) The work must carry prominent notices stating that you modified it, and giving a relevant date.*
> *b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.*
> *c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.*
> *d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.*

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e) Convey the object code using Peer to Peer (P2P) transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d. A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

*A **"User Product"** is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.*

***"Installation Information"** for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.*

If you convey an object code work under this section in, or with, or specifically for use in, a User

Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

***"Additional permissions"** are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.*

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
e) Declining to grant rights under trademark law for use of some trade names, trademarks, or

service marks; or
f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program.

Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

*A **“contributor”** is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's **“contributor version”**.*

*A contributor's **“essential patent claims”** are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, **“control”** includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.*

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

*In the following three paragraphs, a **“patent license”** is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent*

*or covenant not to sue for patent infringement). To **“grant”** such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.*

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED

INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

: **17. Interpretation of Sections 15 and 16.**

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.''
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
```

```
  This is free software, and you are welcome to redistribute it
  under certain conditions; type `show c' for details.
```

The hypothetical commands `show w` *and* `show c` *should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a Graphical User Interface (GUI), you would use an “about box”.*

You should also get your employer (if you work as a programmer) or school, if any, to sign a ***“copyright disclaimer”*** *for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.*

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

End license text

180)

Wikipedia - Comparison of free and open-source software licenses

OSI: The MIT License (MIT)

return to the Open Source Initiative area

Comparison of free and open source licenses[181)]

Table 102: Data Sheet for The MIT License (MIT)

License Attribute	Value
Full Name	The MIT License
SPDX Short Name	MIT
Author	unknown
Latest Version	unknown
Publication Date	unknown
Author URI	unknown
OSI URI	https://opensource.org/licenses/MIT
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

About

Note: The following is an excerpt from the Open Source Initiative web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/MIT

Begin license text.

Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,

INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

End license text.

181)

Wikipedia - Comparison of free and open-source software licenses

OSI: Common Public License, Version 1.0 (CPL-1.0)

return to the Open Source Initiative area

Comparison of free and open source licenses[182)]

Note:

The Open Source Initiative, license steward of the CPL, regards this license as having been superseded by OSI: Eclipse Public License Version 2.0 (EPL-2.0).

Table 103: Data Sheet for Common Public License, Version 1.0 (CPL-1.0)

License Attribute	Value
Full Name	Common Public License
SPDX Short Name	CPL-1.0
Author	IBM
Latest Version	1.0
Publication Date	May 2001
Author URI	http://www.ibm.com/
OSI URI	https://opensource.org/licenses/cpl1.0
Linking	Permissive
Distribution	Permissive
Modification	Permissive
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Permissive
Trademark Grant	No

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/cpl1.0

Begin license text
THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE (“AGREEMENT”). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS
“Contribution” means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
b) in the case of each subsequent Contributor:

i) changes to the Program, and
ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

“Contributor” means any person or entity that distributes the Program.
“Licensed Patents ” mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.
“Program” means the Contributions distributed in accordance with this Agreement.
“Recipient” means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow

Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and
b) its license agreement:

> *i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;*
> *ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;*
> *iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and*
> *iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.*

When the Program is made available in source code form:

> *a) it must be made available under this Agreement; and*
> *b) a copy of this Agreement must be included with each copy of the Program.*

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the

extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of

the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

End License Text

182)

Wikipedia - Comparison of free and open-source software licenses

OSI: Eclipse Public License Version 2.0 (EPL-2.0)

return to the Open Source Initiative area

Comparison of free and open source licenses[183)]

Table 104: Data Sheet for Eclipse Public License Version 2.0 (EPL-2.0)

License Attribute	Value
Full Name	Eclipse Public License Version 2.0
SPDX Short Name	EPL-2.0
Author	Eclipse Foundation
Latest Version	2.0
Publication Date	August 2017
Author URI	https://www.eclipse.org/
OSI URI	https://opensource.org/licenses/EPL-2.0
Linking	Limited
Distribution	Limited
Modification	Limited
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Limited
Trademark Grant	Manually

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. https://opensource.org/licenses/EPL-2.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE (“AGREEMENT”). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS
“Contribution” means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
b) in the case of each subsequent Contributor:

i) changes to the Program, and
ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.
"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.
"Program" means the Contributions distributed in accordance with this Agreement.
"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.
"Derivative Works" shall mean any work, whether in Source Code or other form, that is based on (or derived from) the Program and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship.
"Modified Works" shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.
"Distribute" means the acts of a) distributing or b) making available in any manner that enables the transfer of a copy.
"Source Code" means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.
"Secondary License" means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity.

Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.
d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.
e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. REQUIREMENTS

3.1 If a Contributor Distributes the Program in any form, then:

a) the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and
b) the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:

i) effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
ii) effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
iii) does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and
iv) requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

a) it must be made available under this Agreement, or if the Program (i) is combined with other material in a separate file or files made available under a Secondary License, and (ii) the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and

b) a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability ('notices') contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this

Agreement.

Exhibit A – Form of Secondary Licenses Notice
“This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}.”

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

183)

Wikipedia - Comparison of free and open-source software licenses

OSI: Mozilla Public License (MPL-2.0)

return to the Open Source Initiative area

Comparison of free and open source licenses[184)]

Table 105: Data Sheet for Mozilla Public License (MPL-2.0)

License Attribute	Value
Full Name	Mozilla Public License (MPL-2.0)
SPDX Short Name	MPL-2.0
Author	Mozilla Foundation
Latest Version	2.0
Publication Date	January 2012
Author URI	https://foundation.mozilla.org/
OSI URI	https://opensource.org/licenses/MPL-2.0
Linking	Permissive
Distribution	Copylefted
Modification	Copylefted
Patent Grant	Yes
Private Use	Yes
Sub-licensing	Copylefted
Trademark Grant	No

About

Note: The following is an excerpt from the official License web site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.
https://opensource.org/licenses/MPL-2.0
Excerpt from full license page
2. License Grants and Conditions
2.1. Grants
Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

> *1. under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and*
> *2. under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.*

2.2. Effective Date
The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.
2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

1. *for any code that a Contributor has removed from Covered Software; or*
2. *for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or*
3. *under Patent Claims infringed by Covered Software in the absence of its Contributions.*

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

1. *such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and*
2. *You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.*

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and

the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

184)

Wikipedia - Comparison of free and open-source software licenses

World Wide Web Consortium (W3C)

return to the Technical Standards area

Create a New Page for W3C (e.g. *RDF*) **Acronym** →		Add page

The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web. W3C Mission

Technical Standards

- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- W3C: Decentralized Identifiers (DIDs) 1.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: HTML5 (HTML5)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: XML Path Language (XPath) 3.1

W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification

return to the W3C Standards

Table 106: Data sheet for Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification (W3CAcronym)

Title	Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
Acronym	CSS
Version	2.2
Series	TR
Document Number	
Release Date	12 April 2018
Download	https://www.w3.org/TR/CSS22/css2.pdf

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This specification defines Cascading Style Sheets level 2. CSS is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS simplifies Web authoring and site maintenance.

CSS level 2 supports media-specific style sheets so that authors may tailor the presentation of their documents to visual browsers, aural devices, printers, braille devices, handheld devices, etc. It also supports content positioning, table layout, features for internationalization and some properties related to user interface.

W3C: Decentralized Identifiers (DIDs) 1.0

return to the W3C Standards

Table 107: Data sheet for Decentralized Identifiers (DIDs) V1.0

Title	Decentralized Identifiers (DIDs) v1.0; Core architecture, data model, and representations
Acronym	DID
Version	1.0
Series	TR
Document Number	
Release Date	21 April 2020 – **Working Draft**
Download	https://www.w3.org/TR/2020/WD-did-core-20200421/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. Given the direct call-out to Distributed Ledger Technology (DLT) and Blockchain, have included the first two paragraphs of the Introduction as well.
Note: This specification is under active development; implementers are advised against implementing the specification unless they are directly involved with the W3C DID Working Group.

Abstract

Decentralized identifiers (DIDs) are a new type of identifier that enables verifiable, decentralized digital identity. A DID identifies any subject (e.g., a person, organization, thing, Data Model (DM), abstract entity, etc.) that the controller of the DID decides that it identifies. These new identifiers are designed to enable the controller of a DID to prove control over it and to be implemented independently of any centralized registry, identity provider, or certificate authority. DIDs are URLs that associate a DID subject with a DID document allowing trustable interactions associated with that subject. Each DID document can express cryptographic material, verification methods, or service endpoints, which provide a set of mechanisms enabling a DID controller to prove control of the DID. Service endpoints enable trusted interactions associated with the DID subject. A DID document might contain semantics about the subject that it identifies. A DID document might contain the DID subject itself (e.g. a data model).

This document specifies a common data model, a URL format, and a set of operations for DIDs, DID documents, and DID methods.

Introduction (excerpt)

Conventional identity management systems are based on centralized authorities such as corporate

directory services, certificate authorities, or domain name registries. From the standpoint of cryptographic trust verification, each of these centralized authorities serves as its own root of trust. To make identity management work across these systems requires implementing federated identity management.

The emergence of distributed ledger technology (DLT) and blockchain technology provides the opportunity for fully decentralized identity management. In a decentralized identity system, entities (that is, discrete identifiable units such as, but not limited to, people, organizations, and things) are free to use any shared root of trust. Globally distributed ledgers, decentralized P2P networks, or other systems with similar capabilities, provide the means for managing a root of trust without introducing a centralized authority or a single point of failure. In combination, DLTs and decentralized identity management systems enable any entity to create and manage their own identifiers on any number of distributed, independent roots of trust.

W3C: Document Object Model (DOM) Level 3 Core Specification

return to the W3C Standards

Table 108: Data sheet for Document Object Model (DOM) Level 3 Core Specification

Title	Document Object Model (DOM) Level 3 Core Specification
Acronym	DOM
Version	vvvv
Series	TR
Document Number	
Release Date	7 April 2004
Download	http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/DOM3-Core.pdf

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This specification defines the Document Object Model Core Level 3, a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model Core Level 3 builds on the Document Object Model Core Level 2 [DOM Level 2 Core].

This version enhances DOM Level 2 Core by completing the mapping between DOM and the XML Information Set [XML Information Set], including the support for XML Base [XML Base], adding the ability to attach user information to DOM Nodes or to bootstrap a DOM implementation, providing mechanisms to resolve namespace prefixes or to manipulate “ID” attributes, giving to type information, etc.

Introduction

The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term “document” is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The DOM is designed to be used with any programming language. In order to provide a precise, language-independent specification of the DOM interfaces, we have chosen to define the specifications in Object Management Group (OMG) IDL [185], as defined in the CORBA 2.3.1 specification [CORBA]. In addition to the OMG IDL specification, we provide language bindings for Java [Java] and ECMAScript [ECMAScript] (an industry-standard scripting language based on JavaScript [JavaScript] and JScript [JScript]). Because of language binding restrictions, a mapping has to be applied between the OMG IDL and the programming language in used. For example, while the DOM uses IDL attributes in the definition of interfaces, Java does not allow interfaces to contain attributes.

185)

Note: OMG IDL is used only as a language-independent and implementation-neutral way to specify interfaces. Various other IDLs could have been used ([COM], [Java IDL], [MIDL], ...). In general, IDLs are designed for specific computing environments. The Document Object Model can be implemented in any computing environment, and does not require the object binding runtimes generally associated with such IDLs.

W3C: HTML5 (HTML5)

return to the W3C Standards

Table 109: Data sheet for HTML5

Title	HTML5
Acronym	HTML5
Version	5.3
Series	TR
Document Number	
Release Date	18 October 2018 – **Working Draft**
Download	https://www.w3.org/TR/2018/WD-html53-20181018/

Note: The following is an excerpt from the o W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This specification defines the 5th major version, third minor revision of the core language of the World Wide Web: the Hypertext Markup Language (HTML). In this version, new features continue to be introduced to help Web application authors, new elements continue to be introduced based on research into prevailing authoring practices, and special attention continues to be given to defining clear conformance criteria for user agents in an effort to improve interoperability.

Scope

This specification is limited to providing a semantic-level markup language and associated semantic-level scripting APIs for authoring accessible pages on the Web ranging from static documents to dynamic applications.

The scope of this specification does not include providing mechanisms for media-specific customization of presentation (although default rendering rules for Web browsers are included at the end of this specification, and several mechanisms for hooking into CSS are provided as part of the language).
The scope of this specification is not to describe an entire operating system. In particular, hardware configuration software, image manipulation tools, and applications that users would be expected to use with high-end workstations on a daily basis are out of scope. In terms of applications, this specification is targeted specifically at applications that would be expected to be used by users on an occasional basis, or regularly but from disparate locations, with low Central Processing Unit (CPU) requirements. Examples of such applications include online purchasing systems, searching systems, games (especially multiplayer online games), public telephone books or address books,

communications software (e-mail clients, instant messaging clients, discussion software), document editing software, etc.

W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)

return to the W3C Standards

Table 110: Data sheet for OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)

Title	OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
Acronym	OWL 2
Version	2
Series	TR
Document Number	
Release Date	11 December 2012
Download	http://www.w3.org/2012/pdf/REC-owl2-syntax-20121211.pdf

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 Document Overview describes the overall state of OWL 2, and should be read before other OWL 2 documents.

The meaningful constructs provided by OWL 2 are defined in terms of their structure. As well, a functional-style syntax is defined for these constructs, with examples and informal descriptions. One can reason with OWL 2 ontologies under either the RDF-Based Semantics [OWL 2 RDF-Based Semantics] or the Direct Semantics [OWL 2 Direct Semantics]. If certain restrictions on OWL 2 ontologies are satisfied and the ontology is in OWL 2 DL, reasoning under the Direct Semantics can be implemented using techniques well known in the literature.

Introduction

This document defines the OWL 2 language. The core part of this specification — called the structural specification — is independent of the concrete exchange syntaxes for OWL 2 ontologies. The structural specification describes the conceptual structure of OWL 2 ontologies and thus

provides a normative abstract representation for all (normative and nonnormative) syntaxes of OWL 2. This allows for a clear separation of the essential features of the language from issues related to any particular syntax. Furthermore, such a structural specification of OWL 2 provides the foundation for the implementation of OWL 2 tools such as APIs and reasoners. Each OWL 2 ontology represented as an instance of this conceptual structure can be converted into an RDF graph OWL 2 RDF Mapping; conversely, most OWL 2 ontologies represented as RDF graphs can be converted into the conceptual structure defined in this document as OWL 2 RDF Mapping.

This document also defines the functional-style syntax, which closely follows the structural specification and allows OWL 2 ontologies to be written in a compact form. This syntax is used in the definitions of the semantics of OWL 2 ontologies, the mappings from and into the RDF/XML exchange syntax, and the different profiles of OWL 2. Concrete syntaxes, such as the functional-style syntax, often provide features not found in the structural specification, such as a mechanism for abbreviating IRIs.

Finally, this document defines OWL 2 DL — the subset of OWL 2 with favorable computational properties. Each RDF graph obtained by applying the RDF mapping to an OWL 2 DL ontology can be converted back into the conceptual structure defined in this document by means of the reverse RDF mapping as OWL 2 RDF Mapping.

An OWL 2 ontology is a formal description of a domain of interest. OWL 2 ontologies consist of the following three different syntactic categories:

- *Entities, such as classes, properties, and individuals, are identified by IRIs. They form the primitive terms of an ontology and constitute the basic elements of an ontology. For example, a class a:Person can be used to represent the set of all people. Similarly, the object property a:parentOf can be used to represent the parent-child relationship. Finally, the individual a:Peter can be used to represent a particular person called “Peter”.*
- *Expressions represent complex notions in the domain being described. For example, a class expression describes a set of individuals in terms of the restrictions on the individuals' characteristics.*
- *Axioms are statements that are asserted to be true in the domain being described. For example, using a subclass axiom, one can state that the class a:Student is a subclass of the class a:Person.*

These three syntactic categories are used to express the logical part of OWL 2 ontologies — that is, they are interpreted under a precisely defined semantics that allows useful inferences to be drawn. For example, if an individual a:Peter is an instance of the class a:Student, and a:Student is a subclass of a:Person, then from the OWL 2 semantics one can derive that a:Peter is also an instance of a:Person.

In addition, entities, axioms, and ontologies can be annotated in OWL 2. For example, a class can be given a human-readable label that provides a more descriptive name for the class. Annotations have no effect on the logical aspects of an ontology — that is, for the purposes of the OWL 2 semantics, annotations are treated as not being present. Instead, the use of annotations is left to the applications that use OWL 2. For example, a graphical user interface might choose to visualize a class using one of its labels.

Finally, OWL 2 provides basic support for ontology modularization. In particular, an OWL 2 ontology O can import another OWL 2 ontology O' and thus gain access to all entities, expressions, and axioms in O'.

This document defines the structural specification of OWL 2, the functional syntax for OWL 2, the behavior of datatype maps, and OWL 2 DL. Only the parts of the document related to these three purposes are normative. The examples in this document are informative and any part of the document that is specifically identified as informative is not normative. Further, the informal descriptions of the semantics of OWL 2 constructs in this document are informative; the Direct Semantics and the RDF-Based Semantics are precisely specified in separate documents.

The italicized keywords must, must not, should, should not, and may are used to specify normative features of OWL 2 documents and tools, and are interpreted as specified in RFC 2119.

W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)

return to the W3C Standards

Table 111: Data sheet for RDF 1.1 Concepts and Abstract Syntax (RDF)

Title	RDF 1.1 Concepts and Abstract Syntax
Acronym	RDF
Version	1.1
Series	TR
Document Number	
Release Date	25 February 2014
Download	http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

The Resource Description Framework (RDF) is a framework for representing information in the Web. This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications. The abstract syntax has two key data structures: RDF graphs are sets of subject-predicate-object triples, where the elements may be IRIs, blank nodes, or datatyped literals. They are used to express descriptions of resources. RDF datasets are used to organize collections of RDF graphs, and comprise a default graph and zero or more named graphs. RDF 1.1 Concepts and Abstract Syntax also introduces key concepts and terminology, and discusses datatyping and the handling of fragment identifiers in IRIs within RDF graphs.

W3C: RDF 1.1 Terse RDF Triple Language (Turtle)

return to the W3C Standards

Table 112: Data sheet for RDF 1.1 Turtle

Title	Terse RDF Triple Language
Acronym	Turtle
Version	1.1
Series	TR
Document Number	
Release Date	25 February 2014
Download	http://www.w3.org/TR/2014/REC-turtle-20140225/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web.

This document defines a textual syntax for RDF called Turtle that allows an RDF graph to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. Turtle provides levels of compatibility with the N-Triples [N-TRIPLES] format as well as the triple pattern syntax of the SPARQL W3C Recommendation.

W3C: SPARQL 1.1 Overview (SPARQL)

return to the W3C Standards

Table 113: Data sheet for SPARQL 1.1

Title	SPARQL 1.1
Acronym	SPARQL
Version	1.1
Series	TR
Document Number	
Release Date	21 March 2013
Download	https://www.w3.org/TR/sparql11-overview/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This document is an overview of SPARQL 1.1. It provides an introduction to a set of W3C specifications that facilitate querying and manipulating RDF graph content on the Web or in an RDF store.

W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)

return to the W3C Standards

Table 114: Data sheet for Extensible Markup Language (XML) 1.0 (Fifth Edition) (W3CAcronym)

Title	Extensible Markup Language (XML) 1.0 (Fifth Edition)
Acronym	XML
Version	5
Series	TR
Document Number	
Release Date	26 November 2008
Download	https://www.w3.org/TR/REC-xml/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

Introduction

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language ISO 8879. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

*[Definition: A software module called an **XML processor** is used to read XML documents and provide access to their content and structure.] [Definition: It is assumed that an XML processor is doing its work on behalf of another module, called the **application**.] This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.*

W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures

return to the W3C Standards

Table 115: Data sheet for XML Schema Definition Language (XSD) 1.1 Part 1: Structures)

Title	XML Schema Definition Language (XSD) 1.1 Part 1: Structures
Acronym	XSD
Version	1.1
Series	TR
Document Number	
Release Date	5 April 2012
Download	https://www.w3.org/TR/xmlschema11-1/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This document specifies the XML Schema Definition Language, which offers facilities for describing the structure and constraining the contents of XML documents, including those which exploit the XML Namespace facility. The schema language, which is itself represented in an XML vocabulary and uses namespaces, substantially reconstructs and considerably extends the capabilities found in XML document type definitions (DTDs). This specification depends on XML Schema Definition Language 1.1 Part 2: Datatypes.

Introduction / Purpose

The purpose of XML Schema Definition Language: Structures is to define the nature of XSD schemas and their component parts, provide an inventory of XML markup constructs with which to represent schemas, and define the application of schemas to XML documents.

The purpose of an XSD schema is to define and describe a class of XML documents by using schema components to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements and their content and attributes and their values. Schemas can also provide for the specification of additional document information, such as normalization and defaulting of attribute and element values. Schemas have facilities for self-documentation. Thus, XML Schema Definition Language: Structures can be used to define, describe and catalogue XML vocabularies for classes of XML documents.

Any application that consumes well-formed XML can use the formalism defined here to express syntactic, structural and value constraints applicable to its document instances. The XSD formalism allows a useful level of constraint checking to be described and implemented for a wide spectrum of XML applications. However, the language defined by this specification does not attempt to provide all the facilities that might be needed by applications. Some applications will require constraint capabilities not expressible in this language, and so will need to perform their own additional validations.

W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes

return to the W3C Standards

Table 116: Data sheet for XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes

Title	XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
Acronym	XSD
Version	1.1
Series	TR
Document Number	
Release Date	5 April 2012
Download	https://www.w3.org/TR/xmlschema11-2/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

XML Schema: Datatypes is part 2 of the specification of the XML Schema language. It defines facilities for defining datatypes to be used in XML Schemas as well as other XML specifications. The datatype language, which is itself represented in XML, provides a superset of the capabilities found in XML document type definitions (DTDs) for specifying datatypes on elements and attributes.

W3C: XSL Transformations (XSLT) Version 3.0

return to the W3C Standards

Table 117: Data sheet for XSL Transformations (XSLT) Version 3.0

Title	XSL Transformations (XSLT) Version 3.0
Acronym	XSLT
Version	3.0
Series	TR
Document Number	
Release Date	8 June 2017
Download	https://www.w3.org/TR/2017/REC-xslt-30-20170608/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

This specification defines the syntax and semantics of XSLT 3.0, a language designed primarily for transforming XML documents into other XML documents.

Introduction (What is XSLT?)

This specification defines the syntax and semantics of the XSLT 3.0 language.

A transformation in the XSLT language is expressed in the form of a stylesheet. A stylesheet is made up of one or more well-formed XML documents conforming to the Namespaces in XML Recommendation.

A stylesheet generally includes elements that are defined by XSLT as well as elements that are not defined by XSLT. XSLT-defined elements are distinguished by use of the namespace `http://www.w3.org/1999/XSL/Transform` (see 3.1 XSLT Namespace), which is referred to in this specification as the XSLT namespace. Thus this specification is a definition of the syntax and semantics of the XSLT namespace.

The term stylesheet reflects the fact that one of the important roles of XSLT is to add styling information to an XML source document, by transforming it into a document consisting of XSL formatting objects (see XSL-FO), or into another presentation-oriented format such as HTML, XHTML, or SVG. However, XSLT is used for a wide range of transformation tasks, not exclusively for formatting and presentation applications.

A transformation expressed in XSLT describes rules for transforming input data into output data. The inputs and outputs will all be instances of the XDM data model, described in XDM 3.0. In the simplest and most common case, the input is an XML document referred to as the source tree, and the output is an XML document referred to as the result tree. It is also possible to process multiple source documents, to generate multiple result documents, and to handle formats other than XML. The transformation is achieved by a set of template rules. A template rule associates a pattern, which typically matches nodes in the source document, with a sequence constructor. In many cases, evaluating the sequence constructor will cause new nodes to be constructed, which can be used to produce part of a result tree. The structure of the result trees can be completely different from the structure of the source trees. In constructing a result tree, nodes from the source trees can be filtered and reordered, and arbitrary structure can be added. This mechanism allows a stylesheet to be applicable to a wide class of documents that have similar source tree structures.

Stylesheets have a modular structure; they may contain several packages developed independently of each other, and each package may consist of several stylesheet modules.

W3C: XML Path Language (XPath) 3.1

return to the W3C Standards

Table 118: Data sheet for XML Path Language (XPath) 3.1

Title	XML Path Language (XPath) 3.1
Acronym	Xpath
Version	3.1
Series	TR
Document Number	
Release Date	21 March 2017
Download	https://www.w3.org/TR/2017/REC-xpath-31-20170321/

Note: The following is an excerpt from the W3C site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Abstract

XPath 3.1 is an expression language that allows the processing of values conforming to the Data Model (DM) defined in [XQuery and XPath Data Model (XDM) 3.1]. The name of the language derives from its most distinctive feature, the path expression, which provides a means of hierarchic addressing of the nodes in an XML tree. As well as modeling the tree structure of XML, the data model also includes atomic values, function items, and sequences. This version of XPath supports JSON as well as XML, adding maps and arrays to the data model and supporting them with new expressions in the language and new functions in [XQuery and XPath Functions and Operators 3.1]. These are the most important new features in XPath 3.1.

de facto Standards Bodies

return to Standards Area

A *de facto* Standard is something that is used so widely that it is considered a standard for a given application although it has no official status. [186)] A *de facto* Standard is generally controlled by a corporation (Microsoft, IBM, Google, Amazon, etc) or group (Apache, Linux, Mozilla, etc).

The DIDO RA provides data sheets for de facto standards developed by the following:

Corporate Projects

- Amazon
- Apache Software Foundation (ASF)
- Apple
- Bitcoin
- Consortium for Information & Software Quality (CISQ)
- Ethereum
- Google
- IOTA
- Linux Foundation
- Microsoft
- Oracle
- Talk Openly Develop Openly (TODO)

Individual Projects

- GIT (Revision Control)
- InterPlanetary File System (IPFS)
- Jenkins (Continuous Delivery)
- Jira (Bug tracking system)
- Participating in Open Source Communities
- ZeroMQ Distributed Messaging
- ZeroMQ Message Transport Protocol (ZMTP)

186)

https://whatis.techtarget.com/definition/de-facto-standard

Amazon

return to the de facto Standards area

Amazon is a huge company with an extensive portfolio of de facto standards. Many of them merit little or no interest within the DIDO world; however, Amazon Web Services (AWS) has a lot of functionality deserving consideration by DIDO Communities of Interest (CoI)

- *Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer, including hardware Central Processing Unit (CPU) and Graphics Processing Unit (GPU) for processing; local/RAM memory; hard-disk/SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).https://en.wikipedia.org/wiki/Amazon_Web_Services*

de facto Standards

- None at this time

Guides

The best place to start depends on your level of expertise with Blockchain and AWS — particularly the services related to AWS Blockchain Templates.
https://docs.aws.amazon.com/blockchain-templates/latest/developerguide/what-are-blockchain-templates.html

Proficient with AWS and blockchain

Start with the topic in AWS Blockchain Templates and Features about the framework you want to use. Use the links to launch the AWS Blockchain Template and configure the blockchain network, or download the templates to check them out on your own.

Proficient with AWS and new to blockchain

Start with the Getting Started with AWS Blockchain Templates tutorial. This walks you through creating an introductory Ethereum blockchain network with default settings. When you finish, see AWS Blockchain Templates and Features for an overview of blockchain frameworks and links to learn more about

configuration choices and features.

Beginner with AWS and proficient with blockchain

Start with Setting Up AWS Blockchain Templates. This helps you get set up with fundamentals on AWS, like an account and a user profile. Next, run through the Getting Started with AWS Blockchain Templates tutorial. This tutorial walks you through creating an introductory Ethereum blockchain network. Even if you won't ultimately use Ethereum, you get hands-on experience setting up related services. This experience is useful for all blockchain frameworks. Finally, see the topic in the AWS Blockchain Templates and Features section for your framework.

New to AWS and blockchain

Start with Setting Up AWS Blockchain Templates. This helps you get set up with fundamentals on AWS, like an account and a user profile. Then run through the Getting Started with AWS Blockchain Templates tutorial. This tutorial walks you through creating an introductory Ethereum blockchain network. Take the time to explore the links to learn more about AWS services and Ethereum.

Apache Software Foundation (ASF)

return to the de facto Standards area

Source: About the Apache Software Foundation (ASF)

> *The mission of the Apache Software Foundation (ASF) is to provide software for the public good. We do this by providing services and support for many like-minded software project communities consisting of individuals who choose to participate in ASF activities.*
> *Through the ASF's meritocratic process known as "The Apache Way," more than 730 individual Members and 7,000 Committers successfully collaborate to develop freely available enterprise-grade software, benefiting millions of users worldwide: thousands of software solutions are distributed under the Apache License; and the community actively participates in ASF mailing lists, mentoring initiatives, and ApacheCon, the Foundation's official user conference, trainings, and expo.*

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala

Apache: Log4j

return to the Apache Foundation

> **Note**: The following is an excerpt from the official Apache Log4j site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 119: Data sheet for Log4j

Title	Log4j
Acronym	Log4j
Version	v. 2.11.2
Operating Systems	Java VM
Downloads	https://logging.apache.org/log4j/2.x/download.html
Guidelines	https://logging.apache.org/log4j/2.x/guidelines.html
Supported Languages	Java
Documentation	https://logging.apache.org/log4j/2.x/log4j-users-guide.pdf
License	Apache License, Version 2.0
Reference	https://logging.apache.org/log4j/2.x/index.html

Source: Log4J User's Guide

Introduction

> *Almost every large application includes its own logging or tracing API. In conformance with this rule, the E.U. SEMPER project decided to write its own tracing API. This was in early 1996. After countless enhancements, several incarnations and much work that API has evolved to become log4j, a popular logging package for Java. The package is distributed under the Apache Software License, a fully-fledged open source license certified by the open source initiative. The latest log4j version, including full-source code, class files and documentation can be found at http://logging.apache.org/ log4j/2.x/index.html.*
>
> *Inserting log statements into code is a low-tech method for debugging it. It may also be the only way because debuggers are not always available or applicable. This is usually the case for multithreaded applications and distributed applications at large.*
>
> *Experience indicates that logging was an important component of the development cycle. It offers several advantages. It provides precise context about a run of the application. Once inserted into the code, the generation of logging output requires no human intervention. Moreover, log output can be saved in persistent medium to be studied at a later time. In addition to its use in the development cycle, a sufficiently rich logging package can also be viewed as an auditing tool.*

As Brian W. Kernighan and Rob Pike put it in their truly excellent book “The Practice of Programming”:

> *“As personal choice, we tend not to use debuggers beyond getting a stack trace or the value of a variable or two. One reason is that it is easy to get lost in details of complicated data structures and control flow; we find stepping through a program less productive than thinking harder and adding output statements and self-checking code at critical places. Clicking over statements takes longer than scanning the output of judiciously-placed displays. It takes less time to decide where to put print statements than to single-step to the critical section of code, even assuming we know where that is.”*

More important, debugging statements stay with the program; debugging sessions are transient. Logging does have its drawbacks. It can slow down an application. If too verbose, it can cause scrolling blindness. To alleviate these concerns, log4j is designed to be reliable, fast and extensible. Since logging is rarely the main focus of an application, the log4j API strives to be simple to understand and to use.

Features

Source: log4j Appenders

- Console, directly to the stdout or stderr stream.
- Console, using the PHP echo command.
- A file.
- A file (new file each day).
- A file (new file when a specified size has been reached).
- Sends the log via email. The entire log is sent in one email.
- Sends the log via email. Each log entry is sent in individual emails.
- MongoDB.
- Ignores all log events.
- Database.
- Creates a PHP user-level message using the PHP trigger_error() function.
- A network socket.
- Syslog

Apache: Log4cxx

return to the Apache Foundation

> **Note**: The following is an excerpt from the official Apache Log4cxx site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 120: Data sheet for Log4cxx

Title	Log4cxx
Acronym	Log4cxx
Version	v 0.10.0
Operating Systems	Apache Portable Runtime
Downloads	https://logging.apache.org/log4cxx/latest_stable/download.html
Guidelines	https://logging.apache.org/log4cxx/latest_stable/project-info.html
Supported Languages	C++
Documentation	https://logging.apache.org/log4cxx/latest_stable/apidocs/index.html
License	Apache License, Version 2.0
Reference	https://logging.apache.org/log4cxx/latest_stable/l

Source: Short introduction to Apache log4cxx |

Introduction

> *Apache log4cxx is a logging framework for C++ patterned after Apache log4j, which uses Apache Portable Runtime for most platform-specific code and should be usable on any platform supported by APR. Apache log4cxx is licensed under the Apache License, an open source license certified by the Open Source Initiative.*
>
> *Almost every large application includes its own logging or tracing API. Inserting log statements into code is a low-tech method for debugging it. It may also be the only way because debuggers are not always available or applicable. This is usually the case for multithreaded applications and distributed applications at large.*
>
> *Experience indicates that logging is an important component of the development cycle. It offers several advantages. It provides precise context about a run of the application. Once inserted into the code, the generation of logging output requires no human intervention. Moreover, log output can be saved in persistent medium to be studied at a later time. In addition to its use in the development cycle, a sufficiently rich logging package can also be viewed as an auditing tool.*
>
> *Logging does have its drawbacks. It can slow down an application. If too verbose, it can cause scrolling blindness. To alleviate these concerns, log4cxx is designed to be reliable, fast and extensible. Since logging is rarely the main focus of an application, the log4cxx API strives to be*

simple to understand and to use.

Apache: log4php

return to the Apache Foundation

> **Note**: The following is an excerpt from the official Apache Log4php site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 121: Data sheet for log4php

Title	log4php
Acronym	log4php
Version	v 2.3.0
Operating Systems	PHP runtime
Downloads	https://logging.apache.org/log4php/download.html
Guidelines	https://logging.apache.org/log4php/apidocs/index.html
Supported Languages	PHP
Documentation	https://logging.apache.org/log4php/docs/introduction.html
License	Apache License, Version 2.0
Reference	https://logging.apache.org/log4php/

Source: What is Apache log4php? |

Introduction

Apache log4php™ is a versatile logging framework for PHP.

Feature highlights:
1. Configuration through XML, properties or PHP files
2. Various logging destinations, including:

- *Console (stdout, stderr)*
- *Files (including daily and rolling files)*
- *Email*
- *Databases*
- *Sockets*
- *RFC5424 - The Syslog Protocol (SYSLOG)*

3. Several built-in log message formats, including:

- *HTML*
- *XML*
- *User defined pattern*

4. Nested (NDC) and Mapped (MDC) Diagnostic Contexts.

Apache: log4net

return to the Apache Foundation

> **Note**: The following is an excerpt from the official Apache Log4net site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 122: Data sheet for log4net

Title	log4net
Acronym	log4net
Version	v 2.0.8
Operating Systems	Supported Frameworks
Downloads	https://logging.apache.org/log4net/download_log4net.cgi
Guidelines	https://logging.apache.org/log4net/release/sdk/index.html
Supported Languages	
Documentation	https://logging.apache.org/log4net/release/manual/introduction.html
License	Apache License, Version 2.0
Reference	https://logging.apache.org/log4net/index.html

Introduction

Source: What is Apache Log4net : *The Apache log4net library is a tool to help the programmer output log statements to a variety of output targets. log4net is a port of the excellent Apache log4j™ framework to the Microsoft® .NET runtime. We have kept the framework similar in spirit to the original log4j while taking advantage of new features in the .NET runtime. For more information on log4net see the features document.*

Features

Source: Introduction Manual

- Writes logging events to a database using either prepared statements or stored procedures.
- Writes color highlighted logging events to a an ANSI terminal window.
- Writes logging events to the ASP trace context. These can then be rendered at the end of the ASP page or on the ASP trace page.
- Buffers logging events before forwarding them to child appenders.
- Writes logging events to the application's Console. The events may go to either the standard our stream or the standard error stream. The events may have configurable text and background colors defined for each level.
- Writes logging events to the application's Console. The events may go to either the standard our stream or the standard error stream.
- Writes logging events to the .NET system.
- Writes logging events to the Windows Event Log.

- Writes logging events to a file in the file system.
- Forwards logging events to child appenders.
- Writes logging events to the local SYSLOG service (**UNIX only**).
- Stores logging events in an in memory buffer.
- Writes logging events to the Windows Messenger service. These messages are displayed in a dialog on a users terminal.
- Writes logging events to the debugger. If the application has no debugger, the system debugger displays the string. If the application has no debugger and the system debugger is not active, the message is ignored.
- Writes logging events to a remote SYSLOG service service **using UDP networking**.
- Writes logging events to a remoting sink using .NET remoting.
- Writes logging events to a file in the file system. The RollingFileAppender can be configured to log to multiple files based upon date or file size constraints.
- Sends logging events to an email address.
- Writes SMTP messages as files into a pickup directory. These files can then be read and sent by an SMTP agent such as the IIS SMTP agent.
- Clients connect via Telnet to receive logging events.
- Writes logging events to the .NET trace system.
- Sends logging events as connectionless UDP datagrams to a remote host or a multicast group using a UdpClient.

Apache: log4jscala

return to the Apache Foundation

> **Note**: The following is an excerpt from the official Apache Log4jscala site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 123: Data sheet for log4scala

Title	log4jscala
Acronym	log4jscala
Version	v 2.12
Operating Systems	Java VM
Downloads	https://logging.apache.org/log4j/log4j-scala-11.0/
Guidelines	None
Supported Languages	Scala
Documentation	https://logging.apache.org/log4j/log4j-scala-11.0/log4j-api-scala-sample/index.html
License	Apache License, Version 2.0
Reference	https://logging.apache.org/log4j/log4j-scala-11.0/

Source: Log4j Scala 2.12 API

Introduction

> *Log4j Scala API for Scala 2.12 requires Log4j API, Java 8, the Scala runtime library, and the Scala reflection library. To use Log4j as a logging implementation, then Log4j Core must also be included. Instructions on using these dependencies are in the dependency information page, and additional information is included in the Log4j artifacts page.*

Apple

return to the de facto Standards area

de facto Standards

- Apple: Darwin
- Apple: iOS
- Apple: MacOS

Apple: Darwin

return to Apple

Source: https://9to5mac.com/2014/10/31/apple-releases-os-10-10-yosemite-open-source-darwin-code/

Darwin is an open source UNIX-like computer operating system released by Apple Inc. in 2000. It is composed of code developed by Apple, as well as, code derived from NeXTSTEP, BSD, and other free software projects.[187)]

Darwin forms the core set of components upon which OS X and iOS are based. It is **mostly POSIX compatible**, but has never, by itself, been certified as being compatible with any version of POSIX. (OS X, since Leopard, has been certified as compatible with the Single UNIX Specification version 3 (SUSv3)).

187)
“Apple releases OS X 10.10 Yosemite Open Source Darwin code”, Seth Weintraub, 31 October 2014, https://9to5mac.com/2014/10/31/apple-releases-os-10-10-yosemite-open-source-darwin-code/

Apple: iOS

return to Apple

The iOS kernel is used in the iPhone, iPad and iPod. It uses the XNU[188)] kernel of Darwin [189)].

188)
The OS X kernel. The acronym stands for X is Not Unix. XNU combines the functionality of Mach and BSD with the I/O Kit, the driver model for OS X Apple Developer Glossary: XNU

189)
Another name for the core of the OS X operating system. The Darwin kernel is equivalent to the OS X kernel plus the BSD libraries and commands essential to the BSD command-line environment. Darwin is open source technology. Apple Developer Glossary: Darwin

Apple: MacOS

return to Apple

Source: Mac OS

MacOS is a computer Operating System (OS) for Apple's Macintosh computers. MacOS is based on Apple: Darwin. MacOS (OS X) uses modular design making it easier to add new features. It runs UNIX applications as well as older Mac applications.

Bitcoin

return to the de facto Standards area

Create a New Page for Bitcoin(e.g. *Bitcoinj Developer's Documentation*) **Full Name →**	Add page

> *Bitcoin is a consensus network that enables a new payment system and a completely digital money. It is the first decentralized peer-to-peer payment network that is powered by its users with no central authority or middlemen. From a user perspective, Bitcoin is pretty much like cash for the Internet. Bitcoin can also be seen as the most prominent triple entry bookkeeping system in existence.*[190)]

Participating in Bitcoin Communities

The Where to Discuss Bitcoin is an excellent resource. Many of the avenues open are through social media. Bitcoin has an official forum known as the Bitcointalk Forum.

- Bitcoin: Bitcoinj Developer's Documentation
- Bitcoin: Developer's Guidance
- Bitcoin: Bitcoin Improvement Proposals (BIPs)

190)
"What id Bitcoin? ", https://bitcoin.org/en/faq#what-is-bitcoin

Bitcoin: Bitcoinj Developer's Documentation

return to Bitcoin page

Source: https://bitcoinj.github.io/

What is bitcoinj?

bitcoinj is a library for working with the Bitcoin protocol. It can maintain a wallet, send/receive transactions without needing a local copy of Bitcoin Core and has many other advanced features. It's implemented in Java but can be used from any JVM compatible language: examples in Python and JavaScript are included. It comes with full documentation and many large, well known Bitcoin apps and services are built on it.

javadoc

- Javadoc 0.15.2

Packages

- org.bitcoin
- org.bitcoin.crawler
- org.bitcoin.paymentchannel
- org.bitcoin.protocols.payments
- org.bitcoinj.core
- org.bitcoinj.core.listeners
- org.bitcoinj.crypto
- org.bitcoinj.jni
- org.bitcoinj.kits
- org.bitcoinj.net
- org.bitcoinj.net.discovery
- org.bitcoinj.params
- org.bitcoinj.protocols.channels
- org.bitcoinj.protocols.payments
- org.bitcoinj.script
- org.bitcoinj.signers
- org.bitcoinj.store
- org.bitcoinj.uri
- org.bitcoinj.utils
- org.bitcoinj.wallet
- org.bitcoinj.wallet.listeners

Bitcoin: Developer's Guidance

return to Bitcoin page

> *The Developer Guide aims to provide the information you need to understand Bitcoin and start building Bitcoin-based applications, but it is not a specification. To make the best use of this documentation, you may want to install the current version of Bitcoin Core, either from source or from a pre-compiled executable. Bitcoin Developer's Guide*

Bitcoin: Guide 1 Blockchain

return to Bitcoin Guides

Overview

The block chain provides Bitcoin's public ledger, an ordered and timestamped record of transactions. This system is used to protect against double spending and modification of previous transaction records. Blockchain Guide

Introduction

Each full node in the Bitcoin network independently stores a block chain containing only blocks validated by that node. When several nodes all have the same blocks in their block chain, they are considered to be in consensus. The validation rules these nodes follow to maintain consensus are called consensus rules. This section describes many of the consensus rules used by Bitcoin Core.

Topics

BETA
This documentation uses information provided in Bitcoin Blockchain Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 2 Transactions

return to Bitcoin Guides

Overview

Transactions let users spend satoshis. Each transaction is constructed out of several parts which enable both simple direct payments and complex transactions. Bitcoin Transactions

Introduction

...describe each part and demonstrate how to use them together to build complete transactions.

To keep things simple, this section pretends coinbase transactions do not exist. Coinbase transactions can only be created by Bitcoin miners and they're an exception to many of the rules listed below. Instead of pointing out the coinbase exception to each rule, we invite you to read about coinbase transactions in the block chain section of this guide.

Topics

BETA
This documentation uses information provided in Bitcoin Transactions Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 3 Contracts

return to Bitcoin Guides

Overview

TContracts are transactions which use the decentralized Bitcoin system to enforce financial agreements. Bitcoin contracts can often be crafted to minimize dependency on outside agents, such as the court system, which significantly decreases the risk of dealing with unknown entities in financial transactions. Contracts Guide

Introduction

The following subsections will describe a variety of Bitcoin contracts already in use. Because contracts deal with real people, not just transactions, they are framed below in story format.

Besides the contract types described below, many other contract types have been proposed. Several of them are collected on the Contracts page of the Bitcoin Wiki.

Topics

BETA
This documentation uses information provided in Bitcoin Contracts Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 4 Wallets

return to Bitcoin Guides

Overview

A Bitcoin wallet can refer to either a wallet program or a wallet file. Wallet Guide

Introduction

Wallet programs create public keys to receive satoshis and use the corresponding private keys to spend those satoshis. Wallet files store private keys and (optionally) other information related to transactions for the wallet program.

Wallet programs and wallet files are addressed below in separate subsections, and this document attempts to always make it clear whether we're talking about wallet programs or wallet files.

Topics

- Introductions
- Wallet Programs
- Wallet Files

BETA
This documentation uses information provided in Bitcoin Wallet Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 5 Payment Processing Guide

return to Bitcoin Guides

Overview

Payment processing encompasses the steps spenders and receivers perform to make and accept payments in exchange for products or services. The basic steps have not changed since the dawn of commerce, but the technology has. Payment Processing Guide

Introduction

This section will explain how receivers and spenders can, respectively, request and make payments using Bitcoin—and how they can deal with complications such as refunds and recurrent rebilling.

Topics

BETA
This documentation uses information provided in Bitcoin Payment Processing Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 6 Operating Modes

return to Bitcoin Guides

Overview

The Bitcoin software has different levels of security and tradeoffs in order to verify the blockchain. Operating Modes

Introduction

Currently there are two primary methods of validating the block chain as a client: Full nodes and Simple Payment Verification (SPV) clients. Other methods, such as server-trusting methods, are not discussed as they are not recommended.

Topics

BETA
This documentation uses information provided in Bitcoin Operating Modes Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 7 Peer-to-Peer Networks

return to Bitcoin Guides

Overview

The Bitcoin network protocol allows full nodes (peers) to collaboratively maintain a Peer to Peer (P2P) network for block and transaction exchange. P2P Network Guide

Introduction

Full nodes download and verify every block and transaction prior to relaying them to other nodes. Archival nodes are full nodes which store the entire blockchain and can serve historical blocks to other nodes. Pruned nodes are full nodes which do not store the entire blockchain. Many Simple Payment Verification (SPV) clients also use the Bitcoin network protocol to connect to full nodes.

Consensus rules do not cover networking, so Bitcoin programs may use alternative networks and protocols, such as the high-speed block relay network used by some miners and the dedicated transaction information servers used by some wallets that provide SPV-level security.

To provide practical examples of the Bitcoin peer-to-peer network, this section uses Bitcoin Core as a representative full node and BitcoinJ as a representative SPV client. Both programs are flexible, so only default behavior is described. Also, for privacy, actual IP addresses in the example output below have been replaced with RFC5737 reserved IP addresses.

Topics

BETA
This documentation uses information provided in Bitcoin P@P Network Guide and has not been approved by Bitcoin experts.

Bitcoin: Guide 8 Mining

return to Bitcoin Guides

Introduction

Mining today takes on two forms: Mining Guide

- *Solo mining, where the miner attempts to generate new blocks on his own, with the proceeds from the block reward and transaction fees going entirely to himself, allowing him to receive large payments with a higher variance (longer time between payments)*
- *Pooled mining, where the miner pools resources with other miners to find blocks more often, with the proceeds being shared among the pool miners in rough correlation to the amount of hashing power they each contributed, allowing the miner to receive small payments with a lower variance (shorter time between payments).*

Topics

BETA
This documentation uses information provided in Bitcoin Mining Guide and has not been approved by Bitcoin experts.

Bitcoin: Bitcoin Improvement Proposals (BIPs)

return to Bitcoin page

Bitcoin Improvement Proposal (BIP) is a design document for introducing features or information to Bitcoin. ... This is the standard way of communicating ideas since Bitcoin has no formal structure. The first BIP (BIP 0001) was submitted by Amir Taaki on 2011-08-19 and described what a BIP is. Bitcoin Improvement Proposal (BIP) Source: Bitcoin list of BIPs

Final BIPs

The following non-forking BIPs have been finalized.

- BIP 0011 - M-of-N Standard Transactions
- BIP 0013 - Address Format for pay-to-script-hash
- BIP 0014 - Protocol Version and User Agent
- BIP 0021 - URI Scheme
- BIP 0022 - getblocktemplate - Fundamentals
- BIP 0023 - getblocktemplate - Pooled Mining
- BIP 0031 - Pong message
- BIP 0035 - mempool message
- BIP 0037 - Connection Bloom filtering
- BIP 0061 - Reject P2P message
- BIP 0070 - Payment Protocol
- BIP 0071 - Payment Protocol MIME types
- BIP 0072 - bitcoin: uri extensions for Payment Protocol
- BIP 0073 - Use "Accept" header for response type negotiation with Payment Request URLs
- BIP 0137 - Signatures of Messages using Private Keys
- BIP 0144 - Segregated Witness (Peer Services)
- BIP 0145 - getblocktemplate Updates for Segregated Witness

Forking BIPs

There are three kinds of forks: Temporary [191)], Soft [192)], and Hard[193)]. The following forking BIPs have been finalized.

- BIP 0016 - Pay to Script Hash (soft fork)
- BIP 0030 - Duplicate transactions (soft fork)
- BIP 0034 - Block v2, Height in Coinbase (soft fork)
- BIP 0042 - A finite monetary supply for Bitcoin (soft fork)
- BIP 0065 - OP_CHECKLOCKTIMEVERIFY (soft fork)
- BIP 0068 - Relative lock-time using consensus-enforced sequence numbers (soft fork)
- BIP 0091 - Reduced threshold Segwit MASF (soft fork)
- BIP 0112 - CHECKSEQUENCEVERIFY (soft fork)

- BIP 0113 - Median time-past as endpoint for lock-time calculations (soft fork)
- BIP 0141 - Segregated Witness (Consensus layer) (soft fork)
- BIP 0143 - Transaction Signature Verification for Version 0 Witness Program (soft fork)
- BIP 0147 - Dealing with dummy stack element malleability (soft fork)
- BIP 0148 - Mandatory activation of segwit deployment (soft fork)

191)

Temporary forks are forks that occur when miners, on cryptocurrency systems, discover a block at the same time. This results in two split competing blockchains. Temporary forks are resolved in proof-of-work systems such as Bitcoin when miners select which chain to form subsequent blocks upon. The longest blockchain is viewed as being the 'true' blockchain, and will win out, whilst the shorter chain will be abandoned.
192)

A **soft fork** is a backward compatible method of upgrading a blockchain. In other words, a soft fork is a software upgrade that is backward compatible with previous versions of the software. Soft forks do not require nodes on the network to upgrade to maintain consensus, because all blocks on the soft-forked blockchain follow the old set of consensus rules as well as the new ones.
[[dido:public:ra:xapend.glossary:b:blkchn Soft Fork & Hard Fork Explained]]
193)

A **hard fork** is a permanent divergence from the previous version of a blockchain; a new set of consensus rules are introduced into the network that is not compatible with the older network. In other words, a hard fork can be thought of as a software upgrade that is not compatible with previous versions of the software. All network participants are required to upgrade to the latest version of the software in order to continue verifying and validating new blocks of transactions.

BIP 0011 - M-of-N Standard Transactions

return to the Bitcoin Improvement Proposals

Table 124: Data sheet for M-of-N Standard Transactions

Title	M-of-N Standard Transactions
Layer	Application
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0011
Status	Final
Type	Standards Track
Created	2011-10-18
Post History	2011-10-02
Description	https://github.com/bitcoin/bips/blob/master/bip-0011.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP proposes M-of-N-signatures required transactions as a new 'standard' transaction type.

Motivation

Enable secured wallets, escrow transactions, and other use cases where redeeming funds requires more than a single signature.

A couple of motivating use cases:

- *A wallet secured by a “wallet protection service” (WPS). 2-of-2 signatures required transactions will be used, with one signature coming from the (possibly compromised) computer with the wallet and the second signature coming from the WPS. When sending protected bitcoins, the user's bitcoin client will contact the WPS with the proposed transaction and it can then contact the user for confirmation that they initiated the transaction and that the transaction details are correct. Details for how clients and WPS's communicate are outside the scope of this BIP. Side note: customers should insist that their wallet protection service provide them with copies of the private key(s) used to secure their wallets that they can safely store off-line, so that their coins can be spent even if the WPS goes out of business.*
- *Three-party escrow (buyer, seller and trusted dispute agent). 2-of-3 signatures required transactions will be used. The buyer and seller and agent will each provide a public key, and the buyer will then send coins into a 2-of-3 CHECKMULTISIG transaction and send the seller and the agent the transaction id. The seller will fulfill their obligation and then ask the buyer to co-sign a*

transaction (already signed by seller) that sends the tied-up coins to him (seller).

If the buyer and seller cannot agree, then the agent can, with the cooperation of either buyer or seller, decide what happens to the tied-up coins. Details of how buyer, seller, and agent communicate to gather signatures or public keys are outside the scope of this BIP.

BIP 0013 - Address Format for pay-to-script-hash

return to the Bitcoin Improvement Proposals

Table 125: Data sheet for Address Format for pay-to-script-hash

Title	Address Format for pay-to-script-hash
Layer	Applications
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0013
Status	Final
Type	Standards Track
Created	2011-10-18
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0013.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new type of Bitcoin address to support arbitrarily complex transactions. Complexity in this context is defined as what information is needed by the recipient to respend the received coins, in contrast to needing a single ECDSA private key as in current implementations of Bitcoin.

In essence, an address encoded under this proposal represents the encoded hash of a script, rather than the encoded hash of an ECDSA public key.

Motivation

Enable “end-to-end” secure wallets and payments to fund escrow transactions or other complex transactions. Enable third-party wallet security services.

BIP 0014 - Protocol Version and User Agent

return to the Bitcoin Improvement Proposals

Table 126: Data sheet for Protocol Version and User Agent

Title	Protocol Version and User Agent
Layer	Peer Services
Author	Amir Taaki, Patrick Strateman
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0014
Status	Final
Type	Standards Track
Created	2011-11-10
Post History	2011-11-02
Description	https://github.com/bitcoin/bips/blob/master/bip-0014.mediawiki

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

*** NOTE ***
In this document, bitcoin will be used to refer to the protocol while Satoshi will refer to the current client in order to prevent confusion.

Proposal

> *The version field in `version` and `getblocks` packets will become the protocol version number. The version number in the "blocks" reflects the protocol version from when that block was created.*
>
> *The currently unused `sub_version_num field` in `version` packets will become the new user-agent string.*
>
> *Bitcoin user agents are a modified browser user agent with more structure to aid parsers and provide some coherence. In bitcoin, the software usually works like a stack starting from the core code-base up to the end graphical interface. Therefore the user agent strings codify this relationship.*
>
> ***Basic format:***

```
/Name:Version/Name:Version/.../
```

> ***Example:***

```
    /Satoshi:5.64/bitcoin-qt:0.4/
    /Satoshi:5.12/Spesmilo:0.8/
```

BIP 0021 - URI Scheme

return to the Bitcoin Improvement Proposals

Table 127: Data sheet for URI Scheme

Title	URI Scheme
Layer	Applications
Author	Nils Schneider, Matt Corallo
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0021
Status	Final
Type	Standards Track
Created	2012-09-29
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP proposes a URI scheme for making Bitcoin payments.

Motivation

The purpose of this URI scheme is to enable users to easily make payments by simply clicking links on webpages or scanning QR Codes.

BIP 0022 - getblocktemplate - Fundamentals

return to the Bitcoin Improvement Proposals

Table 128: Data sheet for getblocktemplate - Fundamentals

Title	getblocktemplate - Fundamentals
Layer	API/RPC
Author	Luke Dashjr
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0022
Status	Final
Type	Standards Track
Created	2012-02-28
Post History	
License	BSD-2-Clause
Description	https://github.com/bitcoin/bips/blob/master/bip-0022.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a connivence and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new JSON-RPC method for "smart" Bitcoin miners and proxies. Instead of sending a simple block header for hashing, the entire block structure is sent, and left to the miner to (optionally) customize and assemble.

Copyright

BIP 0023 - getblocktemplate - Pooled Mining

return to the Bitcoin Improvement Proposals

Table 129: Data sheet for getblocktemplate - Pooled Mining

Title	getblocktemplate - Pooled Mining
Layer	API/RPC
Author	Luke Dashjr
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0023
Status	Final
Type	Standards Track
Created	2012-02-28
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0023.mediawiki
Liense	BSD-2-Clause

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes extensions to the getblocktemplate JSON-RPC call to enhance pooled mining.

Copyright

BIP 0031 - Pong message

return to the Bitcoin Improvement Proposals

Table 130: Data sheet for Pong message

Title	Pong message
Layer	Peer Services
Author	Mike Hearn
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0031
Status	Final
Type	Standards Track
Created	2012-04-11
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0031.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document describes a trivial protocol extension that makes it easier for clients to detect dead peer connections.

Motivation

Today there are a few network related problems that can degrade the Bitcoin user experience:

1. Some Bitcoin clients run on platforms that can go to sleep and essentially stop running at any time without warning. Notably, this is very common on both mobiles and laptops (shut the lid). When the system comes back, TCP connections that existed before the sleep still exist but may no longer function correctly, eg, because the IP address has changed, or because the remote peer went away or the connection was timed out by some other system. Currently it can often take a while to notice this has happened.
2. The reference Satoshi client is largely single threaded and when placed under heavy load (e.g., because it is downloading the block chain) becomes very slow to respond to network messages. There's no easy way to detect this has occurred, especially if you are just passively waiting for broadcasts from that peer. A way to detect overloaded remote peers and avoid them would both help balance load and provide a better, more responsive system.
3. When downloading large data structures like the block chain it is efficient to choose a peer that is near to you network-wise, in order to reduce load on often congested trans-national links and

ensure lower latency. Currently it is difficult to measure the latency to a remote peer so clients don't bother, and instead just select a random peer to download from.

All of these can be solved by a backwards compatible protocol modification.

BIP 0035 - mempool message

return to the Bitcoin Improvement Proposals

Table 131: Data sheet for mempool message

Title	mempool message
Layer	Peer Services
Author	Jeff Garzik
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0035
Status	Final
Type	Standards Track
Created	2012-08-16
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0035.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

Make a network node's transaction memory pool accessible via a new “mempool” message. Extend the existing “getdata” message behavior to permit accessing the transaction memory pool.

Motivation

Several use cases make it desirable to expose a network node's transaction memory pool:

1. Simple Payment Verification (SPV) clients, wishing to obtain zero-confirmation transactions sent or received.
2. Miners, to avoid missing lucrative fees, downloading existing network transactions after a restart.
3. Remote network diagnostics.

BIP 0037 - Connection Bloom filtering

return to the Bitcoin Improvement Proposals

Table 132: Data sheet for Connection Bloom filtering

Title	Connection Bloom filtering
Layer	Peer Services
Author	Mike Hearn, Matt Corallo
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0037
Status	Final
Type	Standards Track
Created	2012-10-24
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki
License	PD

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP adds new support to the peer-to-peer protocol that allows peers to reduce the amount of transaction data they are sent. Peers have the option of setting filters on each connection they make after the version handshake has completed. A filter is defined as a Bloom filter on data derived from transactions. A Bloom filter is a probabilistic data structure which allows for testing set membership - they can have false positives but not false negatives.

This document will not go into the details of how Bloom filters work and the reader is referred to Wikipedia for an introduction to the topic.

Motivation

As Bitcoin grows in usage the amount of bandwidth needed to download blocks and transaction broadcasts increases. Clients implementing simplified payment verification do not attempt to fully verify the block chain, instead just checking that block headers connect together correctly and trusting that the transactions in a chain of high difficulty are in fact valid. See the Bitcoin paper for more detail on this mode.

Today, SPV clients have to download the entire contents of blocks and all broadcast transactions, only to throw away the vast majority of the transactions that are not relevant to their wallets. This

slows down their synchronization process, wastes users bandwidth (which on phones is often metered) and increases memory usage. All three problems are triggering real user complaints for the Android "Bitcoin Wallet" app which implements Simple Payment Verification (SPV) mode. In order to make chain synchronization fast, cheap and able to run on older phones with limited memory we want to have remote peers throw away irrelevant transactions before sending them across the network.

BIP 0061 - Reject P2P message

return to the Bitcoin Improvement Proposals

Table 133: Data sheet for Reject P2P message

Title	Reject P2P message
Layer	Peer Services
Author	Gavin Andresen
Comments-Summary	Controversial; some recommendation, and some discouragement.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0061
Status	Final
Type	Standards Track
Created	2014-06-18
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0061.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new message type for the Bitcoin Peer to Peer (P2P) network.

Motivation

Giving peers feedback about why their blocks or transactions are rejected, or why they are being banned for not following the protocol helps interoperability between different implementations.

It also gives Simple Payment Verification (SPV) (simplified payment verification) clients a hint that something may be wrong when their transactions are rejected due to insufficient priority or fees.

BIP 0070 - Payment Protocol

return to the Bitcoin Improvement Proposals

Table 134: Data sheet for Payment Protocol

Title	Payment Protocol
Layer	Applications
Author	Gavin Andresen, Mike Hearn
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0070
Status	Final
Type	Standards Track
Created	2013-07-29
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a protocol for communication between a merchant and their customer, enabling both a better customer experience and better security against man-in-the-middle attacks on the payment process.

Motivation

The current, minimal Bitcoin payment protocol operates as follows:

1. *Customer adds items to an online shopping basket, and decides to pay using Bitcoin.*
2. *Merchant generates a unique payment address, associates it with the customer's order, and asks the customer to pay.*
3. *Customer copies the Bitcoin address from the merchant's web page and pastes it into whatever wallet they are using OR follows a bitcoin: link and their wallet is launched with the amount to be paid.*
4. *Customer authorizes payment to the merchant's address and broadcasts the transaction through the Bitcoin p2p network.*
5. *Merchant's server detects payment and after sufficient transaction confirmations considers the transaction final.*

This BIP extends the above protocol to support several new features:

1. *Human-readable, secure payment destinations– customers will be asked to authorize payment to*

“example.com” instead of an inscrutable, 34-character bitcoin address.

2. *Secure proof of payment, which the customer can use in case of a dispute with the merchant.*
3. *Resistance from man-in-the-middle attacks that replace a merchant's bitcoin address with an attacker's address before a transaction is authorized with a hardware wallet.*
4. *Payment received messages, so the customer knows immediately that the merchant has received, and has processed (or is processing) their payment.*
5. *Refund addresses, automatically given to the merchant by the customer's wallet software, so merchants do not have to contact customers before refunding overpayments or orders that cannot be fulfilled for some reason.*

BIP 0071 - Payment Protocol MIME types

return to the Bitcoin Improvement Proposals

Table 135: Data sheet for Payment Protocol MIME types

Title	Payment Protocol MIME types
Layer	Applications
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0071
Status	Final
Type	Standards Track
Created	2013-07-28
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0071.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP defines a MIME (RFC 2046) Media Type for Bitcoin payment request messages.

Motivation

Wallet or server software that sends payment protocol messages over email or http should follow Internet standards for properly encapsulating the messages.

BIP 0072 - bitcoin: uri extensions for Payment Protocol

return to the Bitcoin Improvement Proposals

Table 136: Data sheet for bitcoin: uri extensions for Payment Protocol

Title	bitcoin: uri extensions for Payment Protocol
Layer	Applications
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0072
Status	Final
Type	Standards Track
Created	2013-07-29
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0072.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes an extension to the bitcoin: URI scheme (BIP 21) to support the payment protocol (BIP 70).

Motivation

Allow users to click on a link in a web page or email to initiate the payment protocol, while being backwards-compatible with existing bitcoin wallets.

BIP 0073 - Use "Accept" header for response type negotiation with Payment Request URLs

return to the Bitcoin Improvement Proposals

Table 137: Data sheet for Use “Accept” header for response type negotiation with Payment Request URLs

Title	Use “Accept” header for response type negotiation with Payment Request URLs
Layer	Applications
Author	Stephen Pair
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0073
Status	Final
Type	Standards Track
Created	2013-08-27
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0073.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes an enhancement to the payment protocol (BIP 70) that addresses the need for short URLs when scanning from QR codes. It generalizes the specification for the behavior of a payment request URL in a way that allows the client and server to negotiate the content of the response using the HTTP Accept: header field. Specifically, the client can indicate to the server whether it prefers to receive a bitcoin URI or a payment request.

Implementation of this BIP does not require full payment request (BIP 70) support.

Motivation

The payment protocol augments the bitcoin: uri scheme with an additional “payment” parameter that specifies a URL where a payment request can be downloaded. This creates long URIs that, when rendered as a QR code, have a high information density. Dense QR codes can be difficult to scan resulting in a more frustrating user experience. The goal is to create a standard that would allow QR scanning wallets to use less dense QR codes. It also makes general purpose QR code scanners more usable with bitcoin accepting websites.

BIP 0137 - Signatures of Messages using Private Keys

return to the Bitcoin Improvement Proposals

Table 138: Data sheet for Signatures of Messages using Private Keys

Title	Signatures of Messages using Private Keys
Layer	Applications
Author	Christopher Gilliard
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0137
Status	Final
Type	Standards Track
Created	2019-02-16
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0137.mediawiki
License	BSD-2-Clause

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document describes a signature format for signing messages with Bitcoin private keys.

The specification is intended to describe the standard for signatures of messages that can be signed and verified between different clients that exist in the field today. Note: that a new signature format has been defined which has a number of advantages over this BIP, but to be backwards compatible with existing implementations this BIP will be useful. See BIP 322 [194] for full details on the new signature scheme.

One of the key problems in this area is that there are several different types of Bitcoin addresses and without introducing specific standards it is unclear which type of address format is being used. See [195]. This BIP will attempt to address these issues and define a clear and concise format for Bitcoin signatures.

Copyright

Motivation

> *Since Bitcoin private keys can not only be used to sign Bitcoin transactions, but also any other message, it has become customary to use them to sign various messages for differing purposes. Some applications of signing messages with a Bitcoin private key are as follows: proof of funds for collateral, credit worthiness, enterence to events, airdrops, audits as well as other applications. While there was no BIP written for how to digitally sign messages with Bitcoin private keys with P2PKH addresses it is a fairly well understood process, however with the introduction of Segwit (both in the form of P2SH and bech32) addresses, it is unclear how to distinguish a P2PKH, P2SH, or bech32 address from one another. This BIP proposes a standard signature format that will allow clients to distinguish between the different address formats.*

194)

https://github.com/bitcoin/bips/blob/master/bip-0322.mediawiki

195)

https://github.com/bitcoin/bitcoin/issues/10542

BIP 0144 - Segregated Witness (Peer Services)

return to the Bitcoin Improvement Proposals

Table 139: Data sheet for Segregated Witness (Peer Services)

Title	Segregated Witness (Peer Services)
Layer	Peer Services
Author	Eric Lombrozo, Pieter Wuille
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0144
Status	Final
Type	Standards Track
Created	2016-01-08
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0144.mediawiki
License	PD

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP defines new messages and serialization formats for propagation of transactions and blocks committing to segregated witness structures.

Motivation

In addition to defining witness structures and requiring commitments in future blocks (BIP141 - Consensus segwit BIP), new mechanisms must be defined to allow peers to advertise support for segregated witness and to relay the witness structures and request them from other peers without breaking compatibility with older nodes.

BIP 0145 - getblocktemplate Updates for Segregated Witness

return to the Bitcoin Improvement Proposals

Table 140: Data sheet for getblocktemplate Updates for Segregated Witness

Title	getblocktemplate Updates for Segregated Witness
Layer	API/RPC
Author	Luke Dashjr
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0145
Status	Final
Type	Standards Track
Created	2016-01-30
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0145.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes modifications to the getblocktemplate JSON-RPC call (BIP22) to support segregated witness as defined by BIP141.

BIP 0016 - Pay to Script Hash (soft fork)

return to the Bitcoin Improvement Proposals

Table 141: Data sheet for Pay to Script Hash

Title	Pay to Script Hash
Layer	Consensus (soft fork)
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0016
Status	Final
Type	Standards Track
Created	2012-01-03
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new “standard” transaction type for the Bitcoin scripting system, and defines additional validation rules that apply only to the new transactions.

Motivation

The purpose of pay-to-script-hash is to move the responsibility for supplying the conditions to redeem a transaction from the sender of the funds to the redeemer.

The benefit is allowing a sender to fund any arbitrary transaction, no matter how complicated, using a fixed-length 20-byte hash that is short enough to scan from a QR code or easily copied and pasted.

BIP 0030 - Duplicate transactions (soft fork)

return to the Bitcoin Improvement Proposals

Table 142: Data sheet for Duplicate transactions

Title	Duplicate transactions
Layer	Consensus (soft fork)
Author	Pieter Wuille
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0030
Status	Final
Type	Standards Track
Created	2012-02-22
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0030.mediawiki
License	BSD-2-Clause

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document gives a specification for dealing with duplicate transactions in the block chain, in an attempt to solve certain problems the reference implementation has with them.

Copyright

Motivation

So far, the Bitcoin reference implementation always assumed duplicate transactions (transactions with the same identifier) didn't exist. This is not true; in particular coinbases are easy to duplicate, and by building on duplicate coinbases, duplicate normal transactions are possible as well. Recently, an attack that exploits the reference implementation's dealing with duplicate transactions was described and demonstrated. It allows reverting fully-confirmed transactions to a single confirmation, making them vulnerable to become unspendable entirely. Another attack is possible that allows forking the block chain for a subset of the network.

BIP 0034 - Block v2, Height in Coinbase (soft fork)

return to the Bitcoin Improvement Proposals

Table 143: Data sheet for Block v2, Height in Coinbase

Title	Block v2, Height in Coinbase
Layer	Consensus (soft fork)
Author	Gavin Andresen
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0034
Status	Final
Type	Standards Track
Created	2012-07-06
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

Bitcoin blocks and transactions are versioned binary structures. Both currently use version 1. This BIP introduces an upgrade path for versioned transactions and blocks. A unique value is added to newly produced coinbase transactions, and blocks are updated to version 2.

Motivation

1. Clarify and exercise the mechanism whereby the bitcoin network collectively consents to upgrade transaction or block binary structures, rules and behaviors
2. Enforce block and transaction uniqueness, and assist unconnected block validation

BIP 0042 - A finite monetary supply for Bitcoin (soft fork)

return to the Bitcoin Improvement Proposals

Table 144: Data sheet for A finite monetary supply for Bitcoin

Title	A finite monetary supply for Bitcoin
Layer	Consensus (soft fork)
Author	Pieter Wuille
Comments-Summary	Summary: Unanimously Recommended for implementation
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0042
Status	Final
Type	Standards Track
Created	2014-04-01
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0042.mediawiki
License	PD

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

Although it is widely believed that Satoshi was an inflation-hating goldbug he never said this, and in fact programmed Bitcoin's money supply to grow indefinitely, forever. He modeled the monetary supply as 4 gold mines being discovered per mibillenium (1024 years), with equal intervals between them, each one being depleted over the course of 140 years.

This poses obvious problems, however. Prominent among them is the discussion on what to call 1 billion Bitcoin, which symbol color to use for it, and when wallet clients should switch to it by default.

To combat this, this document proposes a controversial change: making Bitcoin's monetary supply finite.

Copyright

BIP 0065 - OP_CHECKLOCKTIMEVERIFY (soft fork)

return to the Bitcoin Improvement Proposals

Table 145: Data sheet for OP_CHECKLOCKTIMEVERIFY

Title	OP_CHECKLOCKTIMEVERIFY
Layer	Consensus (soft fork)
Author	Peter Todd
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0065
Status	Final
Type	Standards Track
Created	2014-10-01
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki
License	PD

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new opcode (OP_CHECKLOCKTIMEVERIFY) for the Bitcoin scripting system that allows a transaction output to be made unspendable until some point in the future.

Summary

CHECKLOCKTIMEVERIFY redefines the existing NOP2 opcode. When executed, if any of the following conditions are true, the script interpreter will terminate with an error:

- *the stack is empty; or*
- *the top item on the stack is less than 0; or*
- *the lock-time type (height vs. timestamp) of the top stack item and the `nLockTime` field are not the same; or*
- *the top stack item is greater than the transaction's `nLockTime` field; or*
- *the nSequence field of the `txin` is `0xffffffff`;*

Otherwise, script execution will continue as if a NOP had been executed.

The nLockTime field in a transaction prevents the transaction from being mined until either a certain block height, or block time, has been reached. By comparing the argument to CHECKLOCKTIMEVERIFY against the nLockTime field, we indirectly verify that the desired block height or block time has been reached; until that block height or block time has been reached the transaction output remains

unspendable

BIP 0068 - Relative lock-time using consensus-enforced sequence numbers (soft fork)

return to the Bitcoin Improvement Proposals

Table 146: Data sheet for Relative lock-time using consensus-enforced sequence numbers

Title	Relative lock-time using consensus-enforced sequence numbers
Layer	Consensus (soft fork)
Author	Mark Friedenbach, BtcDrak, Nicolas Dorier, kinoshitajona
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0068
Status	Final
Type	Standards Track
Created	2015-05-28
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP introduces relative lock-time (RLT) consensus-enforced semantics of the sequence number field to enable a signed transaction input to remain invalid for a defined period of time after confirmation of its corresponding outpoint.

Motivation

Bitcoin transactions have a sequence number field for each input. The original idea appears to have been that a transaction in the mempool would be replaced by using the same input with a higher sequence value. Although this was not properly implemented, it assumes miners would prefer higher sequence numbers even if the lower ones were more profitable to mine. However, a miner acting on profit motives alone would break that assumption completely. The change described by this BIP repurposes the sequence number for new use cases without breaking existing functionality. It also leaves room for future expansion and other use cases.

The transaction nLockTime is used to prevent the mining of a transaction until a certain date. nSequence will be repurposed to prevent mining of a transaction until a certain age of the spent output in blocks or timespan. This, among other uses, allows bi-directional payment channels as used in Hashed Timelock Contracts (HTLCs) and BIP112.

BIP 0091 - Reduced threshold Segwit MASF (soft fork)

return to the Bitcoin Improvement Proposals

Table 147: Data sheet for Reduced threshold Segwit MASF

Title	Reduced threshold Segwit MASF
Layer	Consensus (soft fork)
Author	James Hilliard
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0091
Status	Final
Type	Standards Track
Created	2017-05-22
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0091.mediawiki
License	BSD-3-Clause, CC0-1.0

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document specifies a method to activate the existing BIP9 segwit deployment with a majority hashpower less than 95%.

Motivation

Segwit increases the blocksize, fixes transaction malleability, and makes scripting easier to upgrade as well as bringing many other benefits.

This BIP provides a way for a simple majority of miners to coordinate activation of the existing segwit deployment with less than 95% hashpower. For a number of reasons a complete redeployment of segwit is difficult to do until the existing deployment expires. This is due to 0.13.1+ having many segwit related features active already, including all the P2P components, the new network service flag, the witness-tx and block messages, compact blocks v2 and preferential peering. A redeployment of segwit will need to redefine all these things and doing so before expiry would greatly complicate testing.

BIP 0112 - CHECKSEQUENCEVERIFY (soft fork)

return to the Bitcoin Improvement Proposals

Table 148: Data sheet for CHECKSEQUENCEVERIFY

Title	CHECKSEQUENCEVERIFY
Layer	Consensus (soft fork)
Author	BtcDrak, Mark Friedenbach, Eric Lombrozo
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0112
Status	Final
Type	Standards Track
Created	2015-08-10
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki
License	PD

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP describes a new opcode (CHECKSEQUENCEVERIFY) for the Bitcoin scripting system that in combination with BIP 68 allows execution pathways of a script to be restricted based on the age of the output being spent.

Summary

CHECKSEQUENCEVERIFY redefines the existing NOP3 opcode. When executed, if any of the following conditions are true, the script interpreter will terminate with an error:

- *the stack is empty; or*
- *the top item on the stack is less than 0; or*
- *the top item on the stack has the disable flag (1 « 31) unset; and*
  a. the transaction version is less than 2; or
  b. the transaction input sequence number disable flag (1 « 31) is set; or
  c. the relative lock-time type is not the same; or
  d. the top stack item is greater than the transaction input sequence (when masked according to the BIP68);

Otherwise, script execution will continue as if a NOP had been executed.

BIP68 prevents a non-final transaction from being selected for inclusion in a block until the corresponding

input has reached the specified age, as measured in block-height or block-time. By comparing the argument to CHECKSEQUENCEVERIFY against the nSequence field, we indirectly verify a desired minimum age of the the output being spent; until that relative age has been reached any script execution pathway including the CHECKSEQUENCEVERIFY will fail to validate, causing the transaction not to be selected for inclusion in a block.

Motivation

BIP68 repurposes the transaction nSequence field meaning by giving sequence numbers new consensus-enforced semantics as a relative lock-time. However, there is no way to build Bitcoin scripts to make decisions based on this field. By making the nSequence field accessible to script, it becomes possible to construct code pathways that only become accessible some minimum time after proof-of-publication. This enables a wide variety of applications in phased protocols such as escrow, payment channels, or bidirectional pegs.

BIP 0113 - Median time-past as endpoint for lock-time calculations (soft fork)

return to the Bitcoin Improvement Proposals

Table 149: Data sheet for Median time-past as endpoint for lock-time calculations

Title	Median time-past as endpoint for lock-time calculations
Layer	Consensus (soft fork)
Author	Thomas Kerin, Mark Friedenbach
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0113
Status	Final
Type	Standards Track
Created	2015-08-10
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0113.mediawiki
License	PD

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP is a proposal to redefine the semantics used in determining a time-locked transaction's eligibility for inclusion in a block. The median of the last 11 blocks is used instead of the block's timestamp, ensuring that it increases monotonically with each block.

Motivation

At present, transactions are excluded from inclusion in a block if the present time or block height is less than or equal to that specified in the locktime. Since the consensus rules do not mandate strict ordering of block timestamps, this has the unfortunate outcome of creating a perverse incentive for miners to lie about the time of their blocks in order to collect more fees by including transactions that by wall clock determination have not yet matured.

This BIP proposes comparing the locktime against the median of the past 11 block's timestamps, rather than the timestamp of the block including the transaction. Existing consensus rules guarantee this value to monotonically advance, thereby removing the capability for miners to claim more transaction fees by lying about the timestamps of their block.

This proposal seeks to ensure reliable behaviour in locktime calculations as required by BIP65 (CHECKLOCKTIMEVERIFY) and matching the behavior of BIP68 (sequence numbers) and BIP112

(CHECKSEQUENCEVERIFY).

BIP 0141 - Segregated Witness (Consensus layer) (soft fork)

return to the Bitcoin Improvement Proposals

Table 150: Data sheet for Segregated Witness (Consensus layer)

Title	Segregated Witness (Consensus layer)
Layer	Consensus (soft fork)
Author	Eric Lombrozo, Johnson Lau, Pieter Wuille
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0141
Status	Final
Type	Standards Track
Created	2015-12-21
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
License	PD

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This BIP defines a new structure called a “witness” that is committed to blocks separately from the transaction merkle tree. This structure contains data required to check transaction validity but not required to determine transaction effects. In particular, scripts and signatures are moved into this new structure.

The witness is committed in a tree that is nested into the block's existing merkle root via the coinbase transaction for the purpose of making this BIP soft fork compatible. A future hard fork can place this tree in its own branch.

Motivation

The entirety of the transaction's effects are determined by output consumption (spends) and new output creation. Other transaction data, and signatures in particular, are only required to validate the blockchain state, not to determine it.

By removing this data from the transaction structure committed to the transaction merkle tree, several problems are fixed:

1. Nonintentional malleability becomes impossible. Since signature data is no longer part of the transaction hash, changes to how the transaction was signed are no longer relevant to transaction identification. As a solution of transaction malleability, this is superior to the canonical signature

approach (BIP62):

- *It prevents involuntary transaction malleability for any type of scripts, as long as all inputs are signed (with at least one CHECKSIG or CHECKMULTISIG operation)*
- *In the case of an m-of-n CHECKMULTISIG script, a transaction is malleable only with agreement of m private key holders (as opposed to only 1 private key holder with BIP62)*
- *It prevents involuntary transaction malleability due to unknown ECDSA signature malleability*
- *It allows creation of unconfirmed transaction dependency chains without counterparty risk, an important feature for offchain protocols such as the Lightning Network*

2. Transmission of signature data becomes optional. It is needed only if a peer is trying to validate a transaction instead of just checking its existence. This reduces the size of Simple Payment Verification (SPV) proofs and potentially improves the privacy of SPV clients as they can download more transactions using the same bandwidth.
3. Some constraints could be bypassed with a soft fork by moving part of the transaction data to a structure unknown to current protocol, for example:

- *Size of witness could be ignored / discounted when calculating the block size, effectively increasing the block size to some extent*
- *Hard coded constants, such as maximum data push size (520 bytes) or sigops limit could be reevaluated or removed*
- *New script system could be introduced without any limitation from the existing script semantic. For example, a new transaction digest algorithm for transaction signature verification is described in BIP143.*

BIP 0143 - Transaction Signature Verification for Version 0 Witness Program (soft fork)

return to the Bitcoin Improvement Proposals

Table 151: Data sheet for Transaction Signature Verification for Version 0 Witness Program

Title	Transaction Signature Verification for Version 0 Witness Program
Layer	Consensus (soft fork)
Author	Johnson Lau , Pieter Wuille
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0143
Status	Final
Type	Standards Track
Created	2016-01-03
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki
License	PD

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This proposal defines a new transaction digest algorithm for signature verification in version 0 witness program, in order to minimize redundant data hashing in verification, and to cover the input value by the signature.

Motivation

There are 4 ECDSA signature verification codes in the original Bitcoin script system: CHECKSIG, CHECKSIGVERIFY, CHECKMULTISIG, CHECKMULTISIGVERIFY ("sigops"). According to the sighash type (ALL, NONE, SINGLE, ANYONECANPAY), a transaction digest is generated with a double SHA256 of a serialized subset of the transaction, and the signature is verified against this digest with a given public key. The detailed procedure is described in a Bitcoin Wiki article. [196)]

Unfortunately, there are at least 2 weaknesses in the original SignatureHash transaction digest algorithm:

- *For the verification of each signature, the amount of data hashing is proportional to the size of the transaction. Therefore, data hashing grows in O(n2) as the number of sigops in a transaction increases. While a 1 MB block would normally take 2 seconds to verify with an average computer in 2015, a 1MB transaction with 5569 sigops may take 25 seconds to verify. This could be fixed by*

optimizing the digest algorithm by introducing some reusable "midstate", so the time complexity becomes O(n).[197)198)199)]

- *The algorithm does not involve the amount of Bitcoin being spent by the input. This is usually not a problem for online network nodes as they could request for the specified transaction to acquire the output value. For an offline transaction signing device ("cold wallet"), however, the unknowing of input amount makes it impossible to calculate the exact amount being spent and the transaction fee. To cope with this problem a cold wallet must also acquire the full transaction being spent, which could be a big obstacle in the implementation of lightweight, air-gapped wallet. By including the input value of part of the transaction digest, a cold wallet may safely sign a transaction by learning the value from an untrusted source. In the case that a wrong value is provided and signed, the signature would be invalid and no funding might be lost.*[200)]

Deploying the aforementioned fixes in the original script system is not a simple task. That would be either a hardfork, or a softfork for new sigops without the ability to remove or insert stack items. However, the introduction of segregated witness softfork offers an opportunity to define a different set of script semantics without disrupting the original system, as the unupgraded nodes would always consider such a transaction output is spendable by arbitrary signature or no signature at all.[201)]

196)
https://en.bitcoin.it/wiki/OP_CHECKSIG
197)
CVE-2013-2292 https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-2292
198)
New Bitcoin vulnerability: A transaction that takes at least 3 minutes to verify, https://bitcointalk.org/?topic=140078
199)
The Megatransaction: Why Does It Take 25 Seconds?, http://rusty.ozlabs.org/?p=522
200)
SIGHASH_WITHINPUTVALUE: Super-lightweight HW wallets and offline data, https://bitcointalk.org/index.php?topic=181734.0
201)
BIP141: Segregated Witness (Consensus layer), https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

BIP 0147 - Dealing with dummy stack element malleability (soft fork)

return to the Bitcoin Improvement Proposals

Table 152: Data sheet for Dealing with dummy stack element malleability

Title	Dealing with dummy stack element malleability
Layer	Consensus (soft fork)
Author	Johnson Lau
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0147
Status	Final
Type	Standards Track
Created	2016-09-02
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0147.mediawiki
License	PD

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document specifies proposed changes to the Bitcoin transaction validity rules to fix a malleability vector in the extra stack element consumed by `OP_CHECKMULTISIG` and `OP_CHECKMULTISIGVERIFY`.

Motivation

Signature malleability refers to the ability of any relay node on the network to transform the signature in transactions, with no access to the relevant private keys required. For non-segregated witness transactions, signature malleability will change the `txid` and invalidate any unconfirmed child transactions. Although the `txid` of segregated witness (BIP141) transactions is not third party malleable, this malleability vector will change the `wtxid` and may reduce the efficiency of compact block relay (BIP152).

A design flaw in `OP_CHECKMULTISIG` and `OP_CHECKMULTISIGVERIFY` causes them to consume an extra stack element ("dummy element") after signature validation. The dummy element is not inspected in any manner, and could be replaced by any value without invalidating the script. This document specifies a new rule to fix this signature malleability.

BIP 0148 - Mandatory activation of segwit deployment (soft fork)

return to the Bitcoin Improvement Proposals

Table 153: Data sheet for Mandatory activation of segwit deployment

Title	Mandatory activation of segwit deployment
Layer	Consensus (soft fork)
Author	Shaolin Fry
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0148
Status	Final
Type	Standards Track
Created	2017-03-12
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0148.mediawiki
License	BSD-3-Clause, CC0-1.0

Note: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This document specifies a BIP16 like soft fork flag day activation of the segregated witness BIP9 deployment known as "segwit"[202)].

Motivation

Segwit increases the blocksize, fixes transaction malleability, and makes scripting easier to upgrade as well as bringing many other benefits.

It is hoped that miners will respond to this BIP by activating segwit early, before this BIP takes effect. Otherwise this BIP will cause the mandatory activation of the existing segwit deployment before the end of midnight November 15th 2017.

202)

"existing segwit deployment" refers to the BIP9 "segwit" deployment using bit 1, between November 15th 2016 and November 15th 2017 to activate BIP141, BIP143 , and BIP147.

Consortium for Information & Software Quality (CISQ)

return to the de facto Standards area

Create a New Page for CISQ **Full Name** → Add page

The Consortium for Information & Software Quality (CISQ) develops international standards to automate the measurement of software from source code. Industry needs standard, low-cost, automated measures for evaluating software size and structural quality that can be used in controlling the quality, cost, and risk of software that is produced internally or by third parties.

Automation is critical because manual review is infeasible for large multi-layer, multi-language, multi-platform systems. Additionally, DevOps greatly speeds up the deployment of applications, some changing on a daily or even hourly basis, which may result in unintended vulnerabilities without review.https://www.it-cisq.org/standards/

Note: CISQ is an independent program managed by OMG. As a result, not only does it maintain a close working relationship with the OMG, many of its Software Sizing, Code Quality, and Technical Debt standards have been adopted by OMG including one that has been accepted as an ISO standard. More are in progress.

Note: The OMG specifications quoted in this document refer to CISQ as the “Consortium for IT Software Quality”; however, their web site clearly uses the name “Consortium for Information & Software Quality”. Thus, unless quoting from the OMG specifications in Appendix B, this document refers to CISQ using the name shown on its home page.

de facto Standards

Ethereum

return to the de facto Standards area

Source: The following is from the English Wikipedia on Ethereum:

> *Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. It supports a modified version of Nakamoto consensus via transaction-based state transitions.https://en.wikipedia.org/wiki/Ethereum*

> *Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. Like Bitcoin, no one controls or owns Ethereum – it is an open-source project built by many people around the world. But unlike the Bitcoin protocol, Ethereum was designed to be adaptable and flexible. It is easy to create new applications on the Ethereum platform, and with the Homestead release, it is now safe for anyone to use those applications. http://ethdocs.org/en/latest/introduction/what-is-ethereum.html*

Reference: The Ethereum Yellow Paper: https://ethereum.github.io/yellowpaper/paper.pdf

Participating in Ethereum Communities

The Where Can I Join The Ethereum Community? is an excellent resource. Many of the avenues open are through social media. Ethereum has an official forum known as the Ethereum Community Forum.

Create an EIP Data Sheet **0000** (e.g. 0020)→	Add page
Create an EIP Data Sheet **LANGUAGE** (e.g. CPP)	Add page

Ethereum Smart Contract Environment

- Ethereum: Solidity Language Specification
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Ethereum Improvement Proposals (EIPs)
- Ethereum: Clients

Ethereum: Solidity Language Specification

return to the Ethereum Standards

Table 154: Data sheet for Solidity Language Specification

Title	Solidity Language Specification
Acronym	Solidity
Version	0.6.8
Document Number	Solidity
Release Date	17 December 2019
About Specification	https://solidity.readthedocs.io/en/v0.6.8/index.html
Download	https://github.com/ethereum/solidity/releases

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Description

Source: Wikipedia Description of Solidity

Solidity is a statically-typed programming language designed for developing smart contracts that run on the EVM[203)][204)] Solidity is compiled to bytecode that is executable on the EVM. With Solidity, developers are able to write applications that implement self-enforcing business logic embodied in smart contracts, leaving a non-repudiable and authoritative record of transactions.[205)] Writing smart contracts in smart contract specific languages such as Solidity is referred to as easy (ostensibly for those who already have programming skills).[206)]

As specified by Wood it is designed around the ECMAScript syntax to make it familiar for existing web developers;[citation needed] unlike ECMAScript it has static typing and variadic return types. Compared to other EVM-targeting languages of the time such as Serpent and Mutan, Solidity contained a number of important differences. Complex member variables for contracts including arbitrarily hierarchical mappings and structs were supported. Contracts support inheritance, including multiple inheritance with C3 linearization. An application binary interface (ABI) facilitating multiple type-safe functions within a single contract was also introduced (and later supported by Serpent). A documentation system for specifying a user-centric description of the ramifications of a method-call was also included in the proposal, known as "Natural Language Specification".[207)][208)]

Overview

Source: [https://solidity.readthedocs.io/en/v0.5.8/index.html | Ethereum Release Page]]

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.

Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. This is because breaking changes as well as new features and bug fixes are introduced regularly. We currently use a 0.x version number to indicate this fast pace of change.

203)

"Hyperledger Fabric Tutorial - Create a blockchain app for loyalty points". IBM Developer. Retrieved 10 April 2019.

204)

Allison, Ian (30 March 2016). "Microsoft adds Ethereum language Solidity to Visual Studio". International Business Times. Retrieved 11 May 2016.

205)

Allison, Ian (30 March 2016). "Microsoft adds Ethereum language Solidity to Visual Studio". International Business Times. Retrieved 11 May 2016.

206)

Mougayar, William (26 April 2016). The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology. Wiley Publishing. p. 58. ISBN 978-1119300311.

207)

Kapetanios-2008-06-27 & p.309.

208)

ethereum. "Ethereum Natural Specification Format". GitHub.

Ethereum: Ethereum Virtual Machine (EVM)

return to the Ethereum Standards

Table 155: Data sheet for Ethereum Virtual Machine (EVM)

Title	Ethereum Virtual Machine
Acronym	EVM
Version	0.6.8
Document Number	Solidity Smart Contracts: EVM
Release Date	17 December 2019
About Specification	https://solidity.readthedocs.io/en/v0.6.8/introduction-to-smart-contracts.html#index-6
Download	https://github.com/ethereum/solidity/releases

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

The Ethereum Virtual Machine or EVM is the runtime environment for smart contracts in Ethereum. It is not only sandboxed but actually completely isolated, which means that code running inside the EVM has no access to network, filesystem or other processes. Smart contracts even have limited access to other smart contracts.

Accounts

There are two kinds of accounts in Ethereum which share the same address space: External accounts that are controlled by public-private key pairs (i.e. humans) and contract accounts which are controlled by the code stored together with the account.

Transactions

A transaction is a message that is sent from one account to another account (which might be the same or empty, see below). It can include binary data (which is called “payload”) and Ether.

Gas

Upon creation, each transaction is charged with a certain amount of gas, whose purpose is to

limit the amount of work that is needed to execute the transaction and to pay for this execution at the same time. While the EVM executes the transaction, the gas is gradually depleted according to specific rules.

Storage, Memory and the Stack

The Ethereum Virtual Machine has three areas where it can store data- storage, memory and the stack,

Instruction Set

The instruction set of the EVM is kept minimal in order to avoid incorrect or inconsistent implementations which could cause consensus problems. All instructions operate on the basic data type, 256-bit words or on slices of memory (or other byte arrays). For a complete list, please see the list of opcodes as part of the inline assembly documentation. ***REVIEW EVM OPCODES***

Message Calls

Contracts can call other contracts or send Ether to non-contract accounts by the means of message calls. Message calls are similar to transactions, in that they have a source, a target, data payload, Ether, gas and return data. In fact, every transaction consists of a top-level message call which in turn can create further message calls.

Delegatecall / Callcode and Libraries

There exists a special variant of a message call, named delegatecall which is identical to a message call apart from the fact that the code at the target address is executed in the context of the calling contract and msg.sender and msg.value do not change their values.

Logs

It is possible to store data in a specially indexed data structure that maps all the way up to the block level. This feature called logs is used by Solidity in order to implement events.

Deactivate and Self-destruct

The only way to remove code from the blockchain is when a contract at that address performs the selfdestruct operation.

Ethereum: Ethereum Improvement Proposals (EIPs)

return to the Ethereum Standards

Source: Ethereum Improvement Proposals (EIPs)

> *Ethereum Improvement Proposal (EIP) describe standards for the Ethereum platform, including core protocol specifications, client APIs, and contract standards.*

EIP status terms

Draft

an EIP that is open for consideration and is undergoing rapid iteration and changes.

Last Call

an EIP that is done with its initial iteration and ready for review by a wide audience.

Accepted

a core EIP that has been in Last Call for at least 2 weeks and any technical changes that were requested have been addressed by the author. The process for Core Devs to decide whether to encode an EIP into their clients as part of a hard fork is not part of the EIP process. If such a decision is made, the EIP will move to final.

Final (non-Core)

an EIP that has been in Last Call for at least 2 weeks and any technical changes that were requested have been addressed by the author.

Final (Core)

an EIP that the Core Devs have decided to implement and release in a future hard fork or has already been released in a hard fork.

Deferred

an EIP that is not being considered for immediate adoption. May be reconsidered in the future for a subsequent hard fork.

EIP Types

EIPs are separated into a number of types, and each has its own list of EIPs.

Standard Track

Describes any change that affects most or all Ethereum implementations, such as a change to the the network protocol, a change in block or transaction validity rules, proposed application standards/conventions, or any change or addition that affects the interoperability of applications using Ethereum. Furthermore Standard EIPs can be broken down into the following categories.

Core

Improvements requiring a consensus fork (e.g. EIP5, EIP101), as well as changes that are not necessarily consensus critical but may be relevant to "core dev" discussions (for example, the miner/node strategy changes 2, 3, and 4 of EIP86).

Networking

Includes improvements around devp2p (EIP8) and Light Ethereum Subprotocol, as well as proposed improvements to network protocol specifications of whisper and swarm.

Interface

Includes improvements around client API/RPC specifications and standards, and also certain language-level standards like method names (EIP6) and contract ABIs. The label "interface" aligns with the interfaces repo and discussion should primarily occur in that repository before an EIP is submitted to the EIPs repository.

Ethereum Request for Comment (ERC)

Application-level standards and conventions, including contract standards such as token standards (ERC20), name registries (ERC137), URI schemes (ERC681), library/package formats (EIP190), and wallet formats (EIP85).

Meta

Describes a process surrounding Ethereum or proposes a change to (or an event in) a process. Process EIPs are like Standards Track EIPs but apply to areas other than the Ethereum protocol itself. They may propose an implementation, but not to Ethereum's codebase; they often require community consensus; unlike Informational EIPs, they are more than recommendations, and users are typically not free to ignore them. Examples include procedures, guidelines, changes to the decision-making process, and changes to the tools or environment used in Ethereum development. Any meta-EIP is also considered a Process EIP.

Informational

Describes a Ethereum design issue, or provides general guidelines or information to the Ethereum community, but does not propose a new feature. Informational EIPs do not necessarily represent Ethereum community consensus or a recommendation, so users and implementers are free to ignore Informational EIPs or follow their advice.

Final ERCs

Please refer to the Ethereum Improvement Proposals (EIP) Final.

- EIP 20: ERC-20 Token Standard
- EIP 55: Mixed-case checksum address encoding
- EIP 137: Ethereum Domain Name Service - Specification
- EIP 141: Designated invalid EVM instruction
- EIP 155: Simple replay attack protection
- EIP 162: Initial ENS Hash Registrar
- EIP 165: ERC-165 Standard Interface Detection
- EIP 181: ENS support for reverse resolution of Ethereum addresses
- EIP 190: Ethereum Smart Contract Packaging Standard
- EIP 191: Signed Data Standard (DRAFT)
- EIP 211: New opcodes: RETURNDATASIZE and RETURNDATACOPY
- EIP 214: New opcode STATICCALL
- EIP 721: ERC-721 Non-Fungible Token Standard
- EIP 777: ERC-777 Token Standard
- EIP 1167: Minimal Proxy Contract
- EIP 1820: Pseudo-introspection Registry Contract

Interface ERCs

Please refer to the Ethereum Improvement Proposals (EIP) Interface .

- EIP 107: safe "eth_sendTransaction" authorization via html popup (DRAFT)
- EIP 234: `blockHash` to JSON-RPC filter options (DRAFT)

- EIP 695: Create `eth_chainId` method for JSON-RPC (DRAFT)
- EIP 712: Ethereum typed structured data hashing and signing (DRAFT)
- EIP 758: ERC-NN Subscriptions and filters for completed transactions (DRAFT)
- EIP 1102: Opt-in account exposure (DRAFT)
- EIP 1186: RPC-Method to get Merkle Proofs - eth_getProof (DRAFT)
- EIP 1193: Ethereum Provider JavaScript API (DRAFT)
- EIP 1474: Remote Procedure Call (RPC) specification (DRAFT)
- EIP 1767: GraphQL interface to Ethereum node data (DRAFT)
- EIP 1803: ERC-NN Rename opcodes for clarity (DRAFT)
- EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)

EIP 20: ERC-20 Token Standard

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Overview

> *It is the most common and well-known standard within all crypto community. 99% (if not all) issued Initial Coin Offering (ICO) tokens on top of the Ethereum implements this standard. The key benefit we get here is that any application or other smart contracts can interact with a token in a standard manner without a need of knowing other details about the token. Therefore, we have a very pleasant way to create any ICO token and have a standard way to interact with all of them like they are all the same. For instance, crypto wallet developers can avoid custom development and integrations to add new tokens. All they need to know is the Ethereum Token address that implements the standard.*
>
> Source: https://hackernoon.com/5-erc-standards-every-ethereum-developer-should-know-about-c1ea79d3483e

Table 156: Data sheet for ERC-20 Token Standard

Title	ERC-20 Token Standard
Author	Fabian Vogelsteller, Vitalik Buterin
Status	Final
Created	2015-11-19
Description	http://eips.ethereum.org/EIPS/eip-20
Specification	http://eips.ethereum.org/EIPS/eip-20#specification
Category	ERC

Abstract

The following standard allows for the implementation of a standard API for tokens within smart contracts. This standard provides basic functionality to transfer tokens, as well as allow tokens to be approved so they can be spent by another on-chain third party.

EIP 55: Mixed-case checksum address encoding

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 157: Data sheet for Mixed-case checksum address encoding

Title	Mixed-case checksum address encoding
Author	Vitalik Buterin, Alex Van de Sande
Status	Final
Created	2016-01-14
Description	http://eips.ethereum.org/EIPS/eip-55
Specification	http://eips.ethereum.org/EIPS/eip-55#Specification
Category	ERC

Abstract

In English, convert the address to hex, but if the ith digit is a letter (ie. it's one of `abcdef`) print it in uppercase if the 4*ith bit of the hash of the lowercase hexadecimal address is 1 otherwise print it in lowercase.

Rationale

Benefit

- Backwards compatible with many hex parsers that accept mixed case, allowing it to be easily introduced over time
- Keeps the length at 40 characters
- On average there will be 15 check bits per address, and the net probability that a randomly generated address if mistyped will accidentally pass a check is 0.0247%. This is a ~50x improvement over ICAP, but not as good as a 4-byte check code.

EIP 137: Ethereum Domain Name Service - Specification

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 158: Data sheet for Ethereum Domain Name Service - Specification

Title	Ethereum Domain Name Service - Specification
Author	Nick Johnson
Status	Final
Created	2016-04-04
Description	http://eips.ethereum.org/EIPS/eip-137
Specification	http://eips.ethereum.org/EIPS/eip-137#Specification
Category	ERC

Abstract

This draft EIP describes the details of the Ethereum Name Service, a proposed protocol and ABI definition that provides flexible resolution of short, human-readable names to service and resource identifiers. This permits users and developers to refer to human-readable and easy to remember names, and permits those names to be updated as necessary when the underlying resource (contract, content-addressed data, etc) changes.

The goal of domain names is to provide stable, human-readable identifiers that can be used to specify network resources. In this way, users can enter a memorable string, such as 'vitalik.wallet' or 'www.mysite.swarm', and be directed to the appropriate resource. The mapping between names and resources may change over time, so a user may change wallets, a website may change hosts, or a swarm document may be updated to a new version, without the domain name changing. Further, a domain need not specify a single resource; different record types allow the same domain to reference different resources. For instance, a browser may resolve 'mysite.swarm' to the IP address of its server by fetching its A (address) record, while a mail client may resolve the same address to a mail server by fetching its MX (mail exchanger) record.

EIP 141: Designated invalid EVM instruction

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 159: Data sheet for Designated invalid EVM instruction

Title	Designated invalid EVM instruction
Author	Alex Beregszaszi
Status	Final
Created	2017-02-09
Description	http://eips.ethereum.org/EIPS/eip-141
Specification	http://eips.ethereum.org/EIPS/eip-141#Specification
Category	Core

Abstract

An instruction is designated to remain as an invalid instruction.

Motivation

The invalid instruction can be used as a distinct reason to abort execution.

EIP 155: Simple replay attack protection

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 160: Data sheet for Simple replay attack protection

Title	Simple replay attack protection
Author	Vitalik Buterin
Status	Final
Created	2016-10-14
Description	http://eips.ethereum.org/EIPS/eip-155
Specification	http://eips.ethereum.org/EIPS/eip-155#Specification
Category	core

Abstract

Address the problems with Spurious Dragon causing a Hard Fork.

EIP 162: Initial ENS Hash Registrar

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 161: Data sheet for Initial ENS Hash Registrar

Title	Initial ENS Hash Registrar
Author	Maurelian, Nick Johnson, Alex Van de Sande
Status	Final
Created	2016-10-25
Description	http://eips.ethereum.org/EIPS/eip-162
Specification	http://eips.ethereum.org/EIPS/eip-162#Specification
Category	ERC

Abstract

This ERC describes the implementation, as deployed to the main ethereum network on 2017-05-04, of a registrar contract to govern the allocation of names in the Ethereum Name Service (ENS). The corresponding source code is here.

For more background, refer to EIP 137.

> *“Registrars are responsible for allocating domain names to users of the system, and are the only entities capable of updating the ENS; the owner of a node in the ENS registry is its registrar. Registrars may be contracts or externally owned accounts, though it is expected that the root and top-level registrars, at a minimum, will be implemented as contracts.”*

A well designed and governed registrar is essential to the success of the ENS described in EIP 137, but is described separately in this document as it is external to the core ENS protocol.

In order to maximize utility and adoption of a new namespace, the registrar should mitigate speculation and “name squatting”, however the best approach for mitigation is unclear. Thus an “initial” registrar is proposed, which implements a simple approach to name allocation. During the initial period, the available namespace will be significantly restricted to the .eth top level domain, and subdomain shorter than 7 characters in length disallowed. This specification largely describes @alexvandesande and @arachnid’s hash registrar implementation in order to facilitate discussion.

The intent is to replace the Initial Registrar contract with a permanent registrar contract. The Permanent Registrar will increase the available namespace, and incorporate lessons learned from the performance of the Initial Registrar. This upgrade is expected to take place within approximately 2 years of initial deployment.

EIP 165: ERC-165 Standard Interface Detection

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 162: Data sheet for Standard Interface Detection

Title	Standard Interface Detection
Author	Christian Reitwießner, Nick Johnson, Fabian Vogelsteller, Jordi Baylina, Konrad Feldmeier, William Entriken
Status	Final
Created	2018-01-23
Description	http://eips.ethereum.org/EIPS/eip-165
Specification	http://eips.ethereum.org/EIPS/eip-165#Specification
Category	ERC
Requires	EIP 214

Simple Summary

: *Creates a standard method to publish and detect what interfaces a smart contract implements.*

Abstract

> *Herein, we standardize the following:*
>
> 1. *How interfaces are identified*
> 2. *How a contract will publish the interfaces it implements*
> 3. *How to detect if a contract implements ERC-165*
> 4. *How to detect if a contract implements any given interface*

Motivation

> *For some "standard interfaces" like the ERC-20 token interface, it is sometimes useful to query whether a contract supports the interface and if yes, which version of the interface, in order to adapt the way in which the contract is to be interacted with. Specifically for ERC-20, a version identifier has already been proposed. This proposal standardizes the concept of interfaces and standardizes the identification (naming) of interfaces.*

EIP 181: ENS support for reverse resolution of Ethereum addresses

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 163: Data sheet for ENS support for reverse resolution of Ethereum addresses

Title	ENS support for reverse resolution of Ethereum addresses
Author	Nick Johnson
Status	Final
Created	2016-12-01
Description	http://eips.ethereum.org/EIPS/eip-181
Specification	http://eips.ethereum.org/EIPS/eip-181#Specification
Category	ERC

Abstract

This EIP specifies a TLD, registrar, and resolver interface for reverse resolution of Ethereum addresses using ENS. This permits associating a human-readable name with any Ethereum blockchain address. Resolvers can be certain that the reverse record was published by the owner of the Ethereum address in question.

Motivation

While name services are mostly used for forward resolution - going from human-readable identifiers to machine-readable ones - there are many use-cases in which reverse resolution is useful as well:

1. Applications that allow users to monitor accounts benefit from showing the name of an account instead of its address, even if it was originally added by address.
2. Attaching metadata such as descriptive information to an address allows retrieving this information regardless of how the address was originally discovered.
3. Anyone can configure a name to resolve to an address, regardless of ownership of that address. Reverse records allow the owner of an address to claim a name as authoritative for that address.

EIP 190: Ethereum Smart Contract Packaging Standard

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 164: Data sheet for Ethereum Smart Contract Packaging Standard

Title	Ethereum Smart Contract Packaging Standard
Author	Piper Merriam, Tim Coulter, Denis Erfurt, RJ Catalano, Iuri Matias
Status	Final
Created	2017-01-10
Description	http://eips.ethereum.org/EIPS/eip-190
Specification	http://eips.ethereum.org/EIPS/eip-190#Specification
Category	ERC

Abstract

This ERC proposes a specification for Ethereum smart contract packages.

The specification was collaboratively developed by the following Ethereum development framework maintainers.

- *Tim Coulter (Truffle)*
- *Denis Erfurt (Dapple)*
- *Piper Merriam (Populus)*
- *RJ Catalano (Eris PM)*
- *Iuri Matias (Embark)*

Motivation

Packaging is a core piece of modern software development which is missing from the Ethereum ecosystem. The lack of packaging limits the ability for developers to reuse code which negatively affects productivity and security.

A key example of this is the ERC20 standard. There are a few well audited reusable token contracts available but most developers end up writing their own because of the difficulty in finding and reusing existing code.

A packaging standard should have the following positive effects on the ecosystem:

- *Greater overall productivity caused by the ability to reuse existing code.*
- *Increased security caused by the ability to reuse existing well audited implementations of common patterns (ERC20, crowdfunding, etc).*

Smart contract packaging should also have a direct positive effect on the end user. Wallet software will be able to consume a released package and generate an interface for interacting with any deployed contracts included within that package. With the advent of ENS all of the pieces will be in place for a wallet to take a human readable name and present the user with an interface for interacting with the underlying application.

EIP 191: Signed Data Standard (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 165: Data sheet for Signed Data Standard

Title	Signed Data Standard
Author	Martin Holst Swende, Nick Johnson
Status	Draft
Created	2016-01-20
Description	http://eips.ethereum.org/EIPS/eip-191
Specification	http://eips.ethereum.org/EIPS/eip-191#Specification
Category	ERC

Abstract

This ERC proposes a specification about how to handle signed data in Ethereum contracts.

Motivation

Several multisignature wallet implementations have been created which accepts presigned transactions. A presigned transaction is a chunk of binary signed_data, along with signature (r, s and v). The interpretation of the signed_data has not been specified, leading to several problems:

- *Standard Ethereum transactions can be submitted as signed_data. An Ethereum transaction can be unpacked, into the following components: RLP<nonce, gasPrice, startGas, to, value, data> (hereby called RLPdata), r, s and v. If there are no syntactical constraints on signed_data, this means that RLPdata can be used as a syntactically valid presigned transaction.*
- *Multisignature wallets have also had the problem that a presigned transaction has not been tied to a particular validator, i.e a specific wallet. Example:*
  1. Users A, B and C have the 2/3-wallet X
  2. Users A, B and D have the 2/3-wallet Y
  3. User A and B submites presigned transaction to X.
  4. Attacker can now reuse their presigned transactions to X, and submit to Y.

EIP 211: New opcodes: RETURNDATASIZE and RETURNDATACOPY

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 166: Data sheet for New opcodes: RETURNDATASIZE and RETURNDATACOPY

Title	New opcodes: `RETURNDATASIZE` and `RETURNDATACOPY`
Author	Christian Reitwiessner
Status	Final
Created	2017-02-13
Description	http://eips.ethereum.org/EIPS/eip-211
Specification	http://eips.ethereum.org/EIPS/eip-211#Specification
Category	Core
Replaces	EIP 5

Simple Summary / Abstract

A mechanism to allow returning arbitrary-length data inside the EVM has been requested for quite a while now. Existing proposals always had very intricate problems associated with charging gas. This proposal solves the same problem while at the same time, it has a very simple gas charging mechanism and requires minimal changes to the call opcodes. Its workings are very similar to the way `calldata` is handled already; after a call, return data is kept inside a virtual buffer from which the caller can copy it (or parts thereof) into memory. At the next call, the buffer is overwritten. This mechanism is 100% backwards compatible.

Motivation

In some situations, it is vital for a function to be able to return data whose length cannot be anticipated before the call. In principle, this can be solved without alterations to the EVM, for example by splitting the call into two calls where the first is used to compute only the size. All of these mechanisms, though, are very expensive in at least some situations. A very useful example of such a worst-case situation is a generic forwarding contract; a contract that takes call data, potentially makes some checks and then forwards it as is to another contract. The return data should of course be transferred in a similar way to the original caller. Since the contract is generic and does not know about the contract it calls, there is no way to determine the size of the output without adapting the called contract accordingly or trying a logarithmic number of calls.

Compiler implementors are advised to reserve a zero-length area for return data if the size of the return data is unknown before the call and then use `RETURNDATACOPY` in conjunction with `RETURNDATASIZE` to actually retrieve the data.

***Note** this proposal also makes the EIP that proposes to allow to return data in case of an intentional state reversion (EIP-140) much more useful. Since the size of the failure data might be larger than the regular return data (or even unknown), it is possible to retrieve the failure data after the CALL opcode has signalled a failure, even if the regular output area is not large enough to hold the data.*

EIP 214: New opcode STATICCALL

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 167: Data sheet for New opcode STATICCALL

Title	New opcode STATICCALL
Author	Vitalik Buterin, Christian Reitwiessner
Status	Final
Created	2017-02-13
Description	http://eips.ethereum.org/EIPS/eip-214
Specification	http://eips.ethereum.org/EIPS/eip-214#Specification
Category	Core

Simple Summary

To increase smart contract security, this proposal adds a new opcode that can be used to call another contract (or itself) while disallowing any modifications to the state during the call (and its subcalls, if present).

Abstract

This proposal adds a new opcode that can be used to call another contract (or itself) while disallowing any modifications to the state during the call (and its subcalls, if present). Any opcode that attempts to perform such a modification (see below for details) will result in an exception instead of performing the modification.

Motivation

Currently, there is no restriction about what a called contract can do, as long as the computation can be performed with the amount of gas provided. This poses certain difficulties about smart contract engineers; after a regular call, unless you know the called contract, you cannot make any assumptions about the state of the contracts. Furthermore, because you cannot know the order of transactions before they are confirmed by miners, not even an outside observer can be sure about that in all cases.

This EIP adds a way to call other contracts and restrict what they can do in the simplest way. It can be safely assumed that the state of all accounts is the same before and after a static call.

EIP 721: ERC-721 Non-Fungible Token Standard

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 168: Data sheet for Non-Fungible Token Standard

Title	Non-Fungible Token Standard
Author	William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs
Status	Final
Created	2018-01-24
Description	http://eips.ethereum.org/EIPS/eip-721
Specification	http://eips.ethereum.org/EIPS/eip-721#Specification
Category	ERC
Requires	ERC-165

Abstract

The following standard allows for the implementation of a standard API for Non-Fungible Tokens (NFTs) within smart contracts. This standard provides basic functionality to track and transfer NFTs.

We considered use cases of NFTs being owned and transacted by individuals as well as consignment to third party brokers/wallets/auctioneers ("operators"). NFTs can represent ownership over digital or physical assets. We considered a diverse universe of assets, and we know you will dream up many more:

- *Physical property — houses, unique artwork*
- *Virtual collectables — unique pictures of kittens, collectable cards*
- *"Negative value" assets — loans, burdens and other responsibilities*

In general, all houses are distinct and no two kittens are alike. NFTs are distinguishable and you must track the ownership of each one separately.

Motivation

A standard interface allows wallet/broker/auction applications to work with any NFT on Ethereum. We provide for simple ERC-721 smart contracts as well as contracts that track an arbitrarily large number of NFTs. Additional applications are discussed below.

This standard is inspired by the ERC-20 token standard and builds on two years of experience since EIP-20 was created. EIP-20 is insufficient for tracking NFTs because each asset is distinct (non-fungible) whereas each of a quantity of tokens is identical (fungible).

Differences between this standard and EIP-20 are examined below.

EIP 777: ERC-777 Token Standard

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 169: Data sheet for Token Standard

Title	Token Standard
Author	Jacques Dafflon, Jordi Baylina, Thomas Shababi
Status	Final
Created	2017-11-20
Description	http://eips.ethereum.org/EIPS/eip-777
Specification	http://eips.ethereum.org/EIPS/eip-777#Specification
Category	ERC
Requires	1820

Abstract

This standard defines a new way to interact with a token contract while remaining backward compatible with ERC20.

It defines advanced features to interact with tokens. Namely, operators to send tokens on behalf of another address—contract or regular account—and send/receive hooks to offer token holders more control over their tokens.

It takes advantage of ERC1820 to find out whether and where to notify contracts and regular addresses when they receive tokens as well as to allow compatibility with already-deployed contracts.

Motivation

This standard tries to improve upon the widely used ERC20 token standard. The main advantages of this standard are:

1. *Uses the same philosophy as Ether in that tokens are sent with send(dest, value, data).*
1. *Both contracts and regular addresses can control and reject which token they send by registering a `tokensToSend` hook. (Rejection is done by `reverting` in the hook function.)*
1. *Both contracts and regular addresses can control and reject which token they receive by registering a `tokensReceived` hook. (Rejection is done by `reverting` in the hook function.)*
1. *The tokensReceived hook allows to send tokens to a contract and notify it in a single transaction, unlike ERC20 which requires a double call (`approve`/`transferFrom`) to achieve this.*

1. *The holder can “authorize” and “revoke” operators which can send tokens on their behalf. These operators are intended to be verified contracts such as an exchange, a cheque processor or an automatic charging system.*

1. *Every token transaction contains `data` and `operatorData` bytes fields to be used freely to pass data from the holder and the operator, respectively.*

It is backward compatible with wallets that do not contain the `tokensReceived` hook function by deploying a proxy contract implementing the `tokensReceived` hook for the wallet.

EIP 1167: Minimal Proxy Contract

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 170: Data sheet for Minimal Proxy Contract

Title	Minimal Proxy Contract
Author	Peter Murray, Nate Welch, Joe Messerman
Status	Final
Created	2018-06-22
Description	http://eips.ethereum.org/EIPS/eip-1167
Specification	http://eips.ethereum.org/EIPS/eip-1167#Specification
Category	ERC
Requires	ERC 211

Simple Summary

To simply and cheaply clone contract functionality in an immutable way, this standard specifies a minimal bytecode implementation that delegates all calls to a known, fixed address.

Abstract

By standardizing on a known minimal bytecode redirect implementation, this standard allows users and third party tools (e.g. Etherscan) to (a) simply discover that a contract will always redirect in a known manner and (b) depend on the behavior of the code at the destination contract as the behavior of the redirecting contract. Specifically, tooling can interrogate the bytecode at a redirecting address to determine the location of the code that will run - and can depend on representations about that code (verified source, third-party audits, etc). This implementation forwards all calls and 100% of the gas to the implementation contract and then relays the return value back to the caller. In the case where the implementation reverts, the revert is passed back along with the payload data (for revert with message).

Motivation

This standard supports use-cases wherein it is desirable to clone exact contract functionality with a minimum of side effects (e.g. memory slot stomping) and with low gas cost deployment of duplicate proxies.

EIP 1820: Pseudo-introspection Registry Contract

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 171: Data sheet for Pseudo-introspection Registry Contract

Title	Pseudo-introspection Registry Contract
Author	Jordi Baylina, Jacques Dafflon
Status	Final
Created	2019-03-04
Description	http://eips.ethereum.org/EIPS/eip-1820
Specification	http://eips.ethereum.org/EIPS/eip-1820#Specification
Category	ERC
Requires	165 , 214
Replaces	ERC 820

Simple Summary

This standard defines a universal registry smart contract where any address (contract or regular account) can register which interface it supports and which smart contract is responsible for its implementation.

This standard keeps backward compatibility with ERC165.

Abstract

This standard defines a registry where smart contracts and regular accounts can publish which functionality they implement—either directly or through a proxy contract.

Anyone can query this registry to ask if a specific address implements a given interface and which smart contract handles its implementation.

This registry MAY be deployed on any chain and shares the same address on all chains.

Interfaces with zeroes (0) as the last 28 bytes are considered ERC165 interfaces, and this registry SHALL forward the call to the contract to see if it implements the interface.

This contract also acts as an ERC165 cache to reduce gas consumption.

Motivation

There have been different approaches to define pseudo-introspection in Ethereum. The first is ERC165 which has the limitation that it cannot be used by regular accounts. The second attempt is ERC672 which uses reverse Ethereum Name Service (ENS)[209]. Using reverse ENS has two issues. First, it is unnecessarily complicated, and second, ENS is still a centralized contract controlled by a `multisig`. This `multisig` theoretically would be able to modify the system.

This standard is much simpler than ERC672, and it is fully decentralized.

This standard also provides a unique address for all chains. Thus solving the problem of resolving the correct registry address for different chains.

209)

Ethereum Name Servie, https://ens.domains/

EIP 107: safe "eth_sendTransaction" authorization via html popup (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 172: Data sheet for safe “eth_sendTransaction” authorization via html popup

Title	safe “eth_sendTransaction” authorization via html popup
Author	Ronan Sandford
Status	Draft
Created	2016-06-05
Description	http://eips.ethereum.org/EIPS/eip-107
Specification	http://eips.ethereum.org/EIPS/eip-107#Specification
Category	Interface

Abstract

This draft EIP describes the details of an authorization method that if provided by rpc enabled ethereum nodes would allow regular websites to send transactions (via `eth_sendTransaction`) without the need to enable CORS. Instead, user would be asked to confirm the transaction via an html popup.

Every read only rpc call the dapp wants to perform is redirected to an invisible iframe from the node’s domain and for every transaction that the dapp wish to execute, an html popup is presented to the user to allow him/her to cancel or confirm the transaction. This allows the dapp to connect to the node’s rpc api without being granted any kind of privileges. This allows users to safely interact with dapps running in their everyday web browser while their accounts are unlocked. In case the account is not unlocked, and the node has allowed the “personal” api via rpc,the html page also allow the user to enter their password to unlock the account for the scope of the transaction.

Motivation

Currently, if a user navigates to a dapp running on a website using her/his everyday browser, the dapp will by default have no access to the rpc api for security reasons. The user will have to enable CORS for the website’s domain in order for the dapp to work. Unfortunately if the user does so, the dapp will be able to send transactions from any unlocked account without the need for any user consent. In other words, not only does the user need to change the node’s default setting, but the user is also forced to trust the dapp in order to use it. This is of course not acceptable and forces existing dapps to rely on the use of workarounds like:

- *if the transaction is a plain ether transfer, the user is asked to enter it in a dedicated trusted wallet like “Mist”*
- *For more complex case, the user is asked to enter the transaction manually via the node command line interface.*

This proposal aims to provide a safe and user friendly alternative.

Here are some screenshots of the provided implementation of that html popup:

EIP 234: `blockHash` to JSON-RPC filter options (DRAFT)

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 173: Data sheet for Add `blockHash` to JSON-RPC filter options.

Title	Add `blockHash` to JSON-RPC filter options.
Author	Micah Zoltu
Status	Draft
Created	2017-03-24
Description	http://eips.ethereum.org/EIPS/eip-234
Specification	http://eips.ethereum.org/EIPS/eip-234#Specification
Category	Interface

Simple Summary

Add an option to JSON-RPC filter options (used by `eth_newFilter` and `eth_getLogs`) that allows specifying the block hash that should be included in the results. This option would be an alternative to `fromBlock`/`toBlock` options.

Abstract

This addition would allow clients to fetch logs for specific blocks, whether those blocks were in the current main chain or not. This resolves some issues that make it difficult/expensive to author robust clients due to the nature of chain reorgs, unreliable network connections and the result set not containing enough details in the empty case.

EIP 695: Create `eth_chainId` method for JSON-RPC (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 174: Data sheet for Create `eth_chainId` method for JSON-RPC

Title	Create `eth_chainId` method for JSON-RPC
Author	Isaac Ardis, Wei Tang, Fan Torchz
Status	Draft
Created	2017-08-21
Description	http://eips.ethereum.org/EIPS/eip-695
Specification	http://eips.ethereum.org/EIPS/eip-695#Specification
Category	Interface

Simple Summary

> *Include `eth_chainId` method in `eth_-namespaced` JSON-RPC methods.*

Abstract

> *The `eth_chainId` method should return a single STRING result for an integer value in hexadecimal format, describing the currently configured "Chain Id" value used for signing replay-protected transactions, introduced via EIP 155: Simple replay attack protection | EIP-155]].*

Motivation

> *Currently although we can use net_version RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC) call to get the current network ID, there's no RPC for querying the chain ID. This makes it impossible to determine the current actual blockchain using the RPC.*

EIP 712: Ethereum typed structured data hashing and signing (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 175: Data sheet for Ethereum typed structured data hashing and signing

Title	Ethereum typed structured data hashing and signing
Author	Remco Bloemen, Leonid Logvinov, Jacob Evans
Status	Draft
Created	2017-09-12
Description	http://eips.ethereum.org/EIPS/eip-712
Specification	http://eips.ethereum.org/EIPS/eip-712#Specification
Category	Interface
Requires	EIP155, EIP191

Simple Summary

Signing data is a solved problem if all we care about are bytestrings. Unfortunately in the real world we care about complex meaningful messages. Hashing structured data is non-trivial and errors result in loss of the security properties of the system.

As such, the adage “don’t roll your own crypto” applies. Instead, a peer-reviewed well-tested standard method needs to be used. This EIP aims to be that standard.

Abstract

This is a standard for hashing and signing of typed structured data as opposed to just bytestrings. It includes a

- *theoretical framework for correctness of encoding functions,*
- *specification of structured data similar to and compatible with Solidity structs,*
- *safe hashing algorithm for instances of those structures,*
- *safe inclusion of those instances in the set of signable messages,*
- *an extensible mechanism for domain separation,*
- *new RPC call eth_signTypedData, and*
- *an optimized implementation of the hashing algorithm in EVM.*

It does not include replay protection.

EIP 758: ERC-NN Subscriptions and filters for completed transactions (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 176: Data sheet for Subscriptions and filters for completed transactions

Title	Subscriptions and filters for completed transactions
Author	Jack Peterson
Status	Draft
Created	2017-11-09
Description	http://eips.ethereum.org/EIPS/eip-758
Specification	http://eips.ethereum.org/EIPS/eip-758#Specification
Category	Interface
Requires	EIP1474

Simple Summary

> *Provide a way for external callers to be notified of completed transactions, and access the return data of functions executed when a transaction is mined.*

Abstract

> *When a new transaction is submitted successfully to an Ethereum node, the node responds with the transaction's hash. If the transaction involved the execution of a contract function that returns data, the data is discarded. If the return data is state-dependent, which is common, there is no straightforward way for the caller to access or compute the return data. This EIP proposes that callers should be able to subscribe to (or poll for) completed transactions. The Ethereum node sends the return data to the caller when the transactions are sealed.*

Motivation

> *External callers presently have no way of accessing return data from Ethereum, if the function was executed via `eth_sendTransaction` or `eth_sendRawTransaction` RPC request. Access to function return data is in many cases a desirable feature. Making return data available to external callers also addresses the inconsistency between internal callers, which have access to return data within the context of the transaction, and external callers, which do not. Presently, a common workaround is to log the return data, which is bad for several reasons: it contributes to chain bloat, imposes additional gas costs on the caller, and can result in unused logs being written if the*

externally called function involves other (internal) function calls that log their return data. While implementing the original version of this EIP, it was decided to expand this functionality slightly to allow for external callers to be notified of their completed transactions even in the case where there is no return data. This could be either because the method called doesn't return a value, or because the transaction is a simple transfer of value.

EIP 1102: Opt-in account exposure (DRAFT)

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 177: Data sheet for Opt-in account exposure

Title	Opt-in account exposure
Author	Paul Bouchon
Status	Draft
Created	2018-05-04
Description	http://eips.ethereum.org/EIPS/eip-1102
Specification	http://eips.ethereum.org/EIPS/eip-1102#Specification
Category	Interface

Simple summary

This proposal describes a way for DOM environments to expose user accounts in a way that requires user approval.

Abstract

The previous generation of Ethereum-enabled DOM environments follows a pattern of injecting a provider populated with accounts without user consent. This puts users of such environments at risk because malicious websites can use these accounts to view detailed account information and to arbitrarily initiate unwanted transactions on a user's behalf.

This proposal outlines a protocol in which Ethereum-enabled DOM environments expose no accounts until the user approves account access.

EIP 1186: RPC-Method to get Merkle Proofs - eth_getProof (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 178: Data sheet for RPC-Method to get Merkle Proofs - eth_getProof

Title	RPC-Method to get Merkle Proofs - eth_getProof
Author	Simon Jentzsch, Christoph Jentzsch
Status	Draft
Created	2918-06-24
Description	http://eips.ethereum.org/EIPS/eip-1186
Specification	http://eips.ethereum.org/EIPS/eip-1186#Specification
Category	Interface

Simple Summary

> *One of the great features of Ethereum is the fact, that you can verify all data of the state. But in order to allow verification of accounts outside the client, we need an additional function delivering us the required proof. These proofs are important to secure Layer2-Technologies.*

Abstract

> *Ethereum uses a Merkle Tree to store the state of accounts and their storage. This allows verification of each value by simply creating a Merkle Proof. But currently, the standard RPC-Interface does not give you access to these proofs. This EIP suggests an additional RPC-Method, which creates Merkle Proofs for Accounts and Storage Values.*
>
> *Combined with a stateRoot (from the blockheader) it enables offline verification of any account or storage-value. This allows especially IOT-Devices or even mobile apps which are not able to run a light client to verify responses from an untrusted source only given a trusted blockhash.*

Motivation

> *In order to create a MerkleProof access to the full state db is required. The current RPC-Methods allow an application to access single values (`eth_getBalance`, `eth_getTransactionCount`, `eth_getStorageAt`,`eth_getCode`), but it is impossible to read the data needed for a MerkleProof through the standard RPC-Interface. (There are implementations using leveldb and accessing the data via filesystems, but this can not be used for production systems since it*

requires the client to be stopped first - See https://github.com/zmitton/eth-proof)

Today MerkleProofs are already used internally. For example, the Light Client Protocol supports a function creating MerkleProof, which is used in order to verify the requested account or storage-data.

Offering these already existing function through the RPC-Interface as well would enable Applications to store and send these proofs to devices which are not directly connected to the p2p-network and still are able to verify the data. This could be used to verify data in mobile applications or IOT-devices, which are currently only using a remote client.

EIP 1193: Ethereum Provider JavaScript API (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 179: Data sheet for Ethereum Provider JavaScript API

Title	Ethereum Provider JavaScript API
Author	Fabian Vogelsteller, Ryan Ghods, Marc Garreau, Victor Maia
Status	Draft
Created	2018-06-30
Description	http://eips.ethereum.org/EIPS/eip-1193
Specification	http://eips.ethereum.org/EIPS/eip-1193#API
Category	Interface

Summary

This EIP formalizes an Ethereum Provider JavaScript API for consistency across clients and applications. The provider is designed to be minimal and is intended to be available on `window.ethereum` *for cross environment compatibility.*

EIP 1474: Remote Procedure Call (RPC) specification (DRAFT)

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 180: Data sheet for Remote procedure call specification

Title	Remote procedure call specification
Author	Paul Bouchon
Status	Draft
Created	2018-10-02
Description	http://eips.ethereum.org/EIPS/eip-1474
Specification	http://eips.ethereum.org/EIPS/eip-1474#Specification
Category	Interface

Simple Summary

This proposal defines a standard set of remote procedure call methods that an Ethereum node should implement.

Abstract

Nodes created by the current generation of Ethereum clients expose inconsistent and incompatible RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC) methods because no formal Ethereum RPC specification exists. This proposal standardizes such a specification to provide developers with a predictable Ethereum RPC interface regardless of underlying node implementation.

EIP 1767: GraphQL interface to Ethereum node data (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 181: Data sheet for GraphQL interface to Ethereum node data

Title	GraphQL interface to Ethereum node data
Author	Nick Johnson, Raúl Kripalani, Kris Shinn
Status	Draft
Created	2019-02-14
Description	http://eips.ethereum.org/EIPS/eip-1767
Specification	http://eips.ethereum.org/EIPS/eip-1767#Specification
Category	Interface

Abstract

This EIP specifies a GraphQL schema for accessing data stored on an Ethereum node. It aims to provide a complete replacement to the read-only information exposed via the present JSON-RPC interface, while improving on usability, consistency, efficiency, and future-proofing.

Motivation

The current JSON-RPC interface for Ethereum nodes has a number of shortcomings. It's informally and incompletely specified in areas, which has led to incompatibilities around issues such as representation of empty byte strings ("" vs "0x" vs "0x0"), and it has to make educated guesses about the data a user will request, which often leads to unnecessary work.

For example, the `totalDifficulty` field is stored separately from the block header in common Ethereum node implementations, and many callers do not require this field. However, every call to `eth_getBlock` still retrieves this field, requiring a separate disk read, because the RPC server has no way of knowing if the user requires this field or not.

Similarly, transaction receipts in go-ethereum are stored on disk as a single binary blob for each block. Fetching a receipt for a single transaction requires fetching and deserializing this blob, then finding the relevant entry and returning it; this is accomplished by the `eth_getTransactionReceipt` API call. A common task for API consumers is to fetch all the receipts in a block; as a result, node implementations end up fetching and deserializing the same data repeatedly, leading to `O(n^2)` effort to fetch all transaction receipts from a block instead of `O(n)`.

Some of these issues could be fixed with changes to the existing JSON-RPC interface, at the cost of complicating the interface somewhat. Instead, we propose adopting a standard query language,

EIP 1803: ERC-NN Rename opcodes for clarity (DRAFT)

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 182: Data sheet for Rename opcodes for clarity

Title	Rename opcodes for clarity
Author	Alex Beregszaszi
Status	Draft
Created	2017-07-28
Description	http://eips.ethereum.org/EIPS/eip-1803
Specification	http://eips.ethereum.org/EIPS/eip-1803#Specification
Category	Interface
Requires	EIP141

Abstract

Rename the `BALANCE`, `SHA3`, `NUMBER`, `GASLIMIT`, `GAS` and `INVALID` opcodes to reflect their true meaning.

EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 183: Data sheet for Add `blockHash` to JSON-RPC methods which accept a default block parameter

Title	Add `blockHash` to JSON-RPC methods which accept a default block parameter
Author	Charles Cooper
Status	Draft
Created	2019-04-01
Description	http://eips.ethereum.org/EIPS/eip-1898
Specification	http://eips.ethereum.org/EIPS/eip-1898#Specification
Category	Interface

Simple Summary

For JSON-RPC methods which currently accept a default block parameter, additionally allow the parameter to be a block hash.

Abstract

This EIP can be considered a generalization of EIP234. It would enable clients to unambiguously specify the block they want to query for certain JSON-RPC methods, even if the block is not in the canonical chain. This allows clients to maintain a coherent picture of blockchain state that they are interested in, even in the presence of reorgs, without requiring that the node maintain a persistent connection with the client or store any client-specific state.

Ethereum: Clients

return to the Ethereum Standards

Source: What exactly is an Ethereum Client

An 'Ethereum client' is a term referring to any node able to parse and verify the blockchain, its smart contracts and everything related. It also allows you/provides interfaces to create transactions and mine blocks which is the key for any blockchain interaction.

Official reference implementations (CLI)

There are currently three reference Command Line Interfaces (CLI) implementations available, as you already highlighted:

- eth - C++ client of the webthree project. It was formerly known as cpp-ethereum: https://github.com/ethereum/webthree-umbrella
- geth - Golang client of the go-ethereum project: https://github.com/ethereum/go-ethereum
- pyethapp - Python client of the pyethereum project: https://github.com/ethereum/pyethapp

All clients should work the same, from the user's perspective. They provide the same interfaces and so on. For example, if you launch a DApp or the Ethereum Wallet or a DApp browser instance, it should not note any difference in communicating with the client.

Official reference implementations (GUI)

Graphical clients available to the Ethereum core developers are:

- mist which works on top of geth or eth and aims to be a DApp browser and currently implements the ethereum-wallet-dapp. https://github.com/ethereum/mist
- alethzero is internally called the hardcore client but it's being deprecated. https://github.com/ethereum/alethzero

Third party implementations (CLI)

Non-official clients implementing the yellow paper specification are:

- parity - Rust client by ethcore: https://github.com/paritytech/parity
- ethereumj - Java client by the ether.camp team: https://github.com/ethereum/ethereumj
- ethereumjs-vm - Ethereum Virtual Machine in Javascript: https://github.com/ethereumjs/ethereumjs-vm
- ethereumH - Haskell client from consensys, but it's not developed anymore: https://github.com/jamshidh/ethereum-client-haskell
- ruby-ethereum - Ruby client: https://github.com/janx/ruby-ethereum

- node-blockchain-server - simple Javascript server: https://github.com/ethereumjs/node-blockchain-server
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project

Ethereum: cpp Project

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 184: Data sheet for Ethereum cpp Project

Title	Ethereum cpp Project
Language	cpp
Created	2016
Repository	https://github.com/ethereum/cpp-ethereum/
Description	http://www.ethdocs.org/en/latest/ethereum-clients/cpp-ethereum/index.html
Category	Client

Abstract

Provides an API for C++ (cpp) on a wide range of operating systems and devices:

Operating Systems

- Linux
- BSD
- OS X
- Windows

Devices

- All varieties of desktop and laptop devices (Windows, OS X, Desktop Linux)
  a. 64-bit (with rebuilt binaries)
  b. 32-bit (not officially supported, but they work)
- Smartphones
- Linux
- Single Board Computers (i.e., Raspberry PI)

Source: Portability

Ethereum: Ethereumh Project

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference. Also – the links provided may not result in anything.

Table 185: Data sheet for Ethereum Ethereumh Project

Title	Ethereum Ethereumh Project
Language	Haskell
Created	2016
Repository	https://github.com/blockapps/ethereumH
Description	http://www.ethdocs.org/en/latest/ethereum-clients/ethereumh/index.html
Category	Client

Abstract

Source: Ethereumh Project

> *This package provides a tool written in Haskell to allow you to connect to the Ethereum blockchain*

Ethereum: Ethereumjs-lib Project

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 186: Data sheet for Ethereum Ethereumjs-lib Project

Title	Ethereum Ethereumjs-lib Project
Language	Javascript
Created	2016
Repository	https://github.com/ethereumjs/ethereumjs-lib
Description	http://www.ethdocs.org/en/latest/ethereum-clients/ethereumjs-lib/index.html
Category	Client

Abstract

Ethereumjs-lib is the javascript library of core Ethereum functions as described in the Yellow Paper[210]. This is a simple meta-module that provides the following modules. Most JS modules are tracked in ethereumjs

- *VM - The Ethereum virtual machine and state processing functions*
- *Blockchain - Blockchain management*
- *Block - Block Schema definition and validation*
- *Transaction - Transaction Schema definition and validation*
- *Account - Account Schema definition and validation*
- *rlp - Recursive Length Prefix serialization*
- *Trie - Modified Merkle Patricia Tree*
- *Ethash - Ethereum's Proof of Work algorithm*
- *utils - Miscellaneous helper functions*
- *devp2p - The networking protocol*
- *devp2p-dpt - The disputed peer table*

210)

"ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER BYZANTIUM VERSION 3e36772 - 2019-05-12", https://ethereum.github.io/yellowpaper/paper.pdf

Ethereum: Ethereum_j Project

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 187: Data sheet for Ethereum Ethereum_j Project

Title	Ethereum Ethereum_j Project
Language	Java
Created	2016
Repository	https://github.com/ethereum/ethereumj
Description	http://www.ethdocs.org/en/latest/ethereum-clients/ethereumj/index.html
Category	Client

Abstract

Source: Ethereum_j Project

Ethereum(J) is a pure-Java implementation of the Ethereum protocol. It is provided as a library that can be embedded in any Java/Scala project and to provide full support for Ethereum protocol and sub-services. Ethereum(J) was first developed by Roman Mandeleil and is now sponsored by <ether.camp>.

Ethereum(J) supports Central Processing Unit (CPU) mining. It is currently implemented in pure Java and can be used in private and test networks. You may even mine on the live Ethereum network, even though it is not economically feasible.

Ethereum: Go-ethereum Project

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 188: Data sheet for Go-ethereum

Title	Go-ethereum
Lnaguage	Go
Status	Draft
Created	2016
Repository	https://github.com/ethereum/go-ethereum
Description	http://www.ethdocs.org/en/latest/ethereum-clients/go-ethereum/index.html
Category	Client

Abstract

Source: go-ethereum Project

*The go-ethereum client is commonly referred to as **geth**, which is the the command line interface for running a full ethereum node implemented in Go. By installing and running geth, you can take part in the ethereum frontier live network and:*

- *mine real ether*
- *transfer funds between addresses*
- *create contracts and send transactions*
- *explore block history*
- *and much much more*

Ethereum: Parity Project

Return to Ethereum ERCs

> **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 189: Data sheet for Ethereum Parity Project

Title	Ethereum Parity Project
Language	Rust
Created	2016
Repository	https://github.com/ethcore/parity
Description	http://www.ethdocs.org/en/latest/ethereum-clients/parity/index.html
Category	Client

Abstract

Source: Parity Project

> *Parity claims to be the world's fastest and lightest Ethereum client. It is written in the Rust language, which offers improved reliability, performance, and code clarity. Parity is being developed by Ethcore, which was founded by several members of the Ethereum Foundation.*

Ethereum: Pyethapp Project

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 190: Data sheet for Ethereum Pyethapp Project

Title	Ethereum Pyethapp Project
Created	2016
Repository	https://github.com/ethereum/pyethapp
Description	http://www.ethdocs.org/en/latest/ethereum-clients/go-ethereum/index.html
Category	Client

Abstract

Source: Pyethapp Project

pyethapp *is the python-based client implementing the Ethereum cryptoeconomic state machine. The python implementation aims to provide an easily hackable and extendable codebase.*

pyethapp leverages two ethereum core components to implement the client:

- *pyethereum - the core library, featuring the blockchain, the ethereum virtual machine, mining*
- *pydevp2p - the p2p networking library, featuring node discovery for and transport of multiple services over multiplexed and encrypted connections*

Ethereum: Ruby-ethereum Project

Return to Ethereum ERCs

Note: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 191: Data sheet for Ethereum Ruby-ethereum Project

Title	Ethereum Ruby-ethereum Project
Language	Ruby
Created	2016
Repository	https://github.com/janx/ruby-ethereum
Description	http://www.ethdocs.org/en/latest/ethereum-clients/ruby-ethereum/index.html
Category	Client

Abstract

Source: Ruby-ethereum Project

*ruby-ethereum is an implementation of the **Ethereum Virtual Machine** written in Ruby.*

Related Projects

- ruby-serpent: Ruby binding to the Ethereum Serpent compiler.
- ethereum-ruby: a pure-Ruby JSON-RPC wrapper for communicating with an Ethereum node.

Google

return to the de facto Standards area

Create a New Page for Google **Full Name** → Add page

de facto Standards

- Google: Android
- Google: Go (software language)
- Google: gRPC
- Google: Protocol Buffers

Google: Android

return to Google page

Source: The following is from the English Wikipedia on Android

> ***Android*** *is a mobile operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software, and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.*
>
> *Android is also associated with a suite of proprietary software developed by Google, called Google Mobile Services (GMS),*[211] *that frequently comes pre-installed on devices. This includes core apps such as Gmail, the application store/digital distribution platform Google Play and associated Google Play Services development platform, and usually includes the Google Chrome web browser and Google Search app. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems such as Amazon.com's Fire OS, which use their own equivalents to Google Mobile Services.*

Table 192: Data Sheet for Jenkins.

Characteristic	Value
Developer	Google, Open Handset Alliance
Written in	Java (UI), C (core), C++ and others [212]
OS family	Unix-like (Modified Linux kernel)
Working state	Current
Source model	Open source (most devices include proprietary components, such as Google Play)
Initial release	September 23, 2008; 10 years ago Morrill, Dan (September 23, 2008). [213]
Latest release	9.0 “Pie” / August 6, 2018
Latest preview	Android Q Beta 4 (QPP4.190502.018)[214] / June 5, 2019;
Marketing target	Smartphones, tablet computers, smartTVs (Android TV), Android Auto and smartwatches (Wear OS)
Available in	100+ languages[215]
Update method	Over-the-air
Package manager	APK (primarily through Google Play; installation of APKs also possible locally or from alternative sources such as F-Droid)
Platforms	32- and 64-bit ARM, x86 and x86-64
Kernel type	Monolithic
Userland	Bionic libc,[216] mksh shell,[217] Toybox as core utilities (beginning with Android 6.0)[218][219]
Default user interface	Graphical (multi-touch)
License	Apache License 2.0, GNU GPL v2 for the Linux kernel modifications[220]

Characteristic	Value
Official website	http://www.android.com

211)

Android is also associated with a suite of proprietary software developed by Google, called Google Mobile Services (GMS), that frequently comes pre-installed on devices. This includes core apps such as Gmail, the application store/digital distribution platform Google Play and associated Google Play Services development platform, and usually includes the Google Chrome web browser and Google Search app. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems such as Amazon.com's Fire OS, which use their own equivalents to Google Mobile Services.

212)

“Android Language Breakdown”. Open Hub. October 25, 2017. Archived from the original on December 14, 2017. Retrieved December 15, 2017.

213)

"Announcing the Android 1.0 SDK, release 1". Android Developers Blog. Google. Archived from the original on March 5, 2017. Retrieved March 11, 2017.

214)

"Support and Release Notes". Android Developers Blog. Google. Retrieved May 7, 2019.

215)

"Android 7.0 Nougat". Archived from the original on August 22, 2016.

216)

“android/platform/bionic/”. Archived from the original on December 3, 2017.https://android.googlesource.com/platform/bionic/+/master/libc/

217)

“android/platform/external/mksh/”. Archived from the original on January 21, 2016.https://web.archive.org/web/20160121112754/https://android.googlesource.com/platform/external/mksh/%2B/master

218)

“android/platform/external/toybox/toys/”. Archived from the original on March 14, 2016.https://android.googlesource.com/platform/external/toybox/+/master/toys/

219)

“Android gets a toybox”. Archived from the original on March 4, 2016.https://lwn.net/Articles/629362/

220)

Licenses“. Android Source. Google. Archived from the original on December 15, 2016. Retrieved March 11, 2017.https://source.android.com/source/licenses.html

Google: Go (software language)

return to Google page

Source: The following is from the ''Golang.ord'' site

> *The language is called Go. The “golang” moniker arose because the web site is golang.org, not go.org, which was not available to us. Many use the golang name, though, and it is handy as a label. For instance, the Twitter tag for the language is “#golang”. The language's name is just plain Go, regardless.*
>
> ***Note:** Although the official logo has two capital letters, the language name is written Go, **not** **GO**.*
>
> *Go was born out of frustration with existing languages and environments for the work we were doing at Google. Programming had become too difficult and the choice of languages was partly to blame. One had to choose either efficient compilation, efficient execution, or ease of programming; all three were not available in the same mainstream language. Programmers who could were choosing ease over safety and efficiency by moving to dynamically typed languages such as Python and JavaScript rather than C++ or, to a lesser extent, Java.*
>
> *We were not alone in our concerns. After many years with a pretty quiet landscape for programming languages, Go was among the first of several new languages—Rust, Elixir, Swift, and more—that have made programming language development an active, almost mainstream field again.*
>
> *Go addressed these issues by attempting to combine the ease of programming of an interpreted, dynamically typed language with the efficiency and safety of a statically typed, compiled language. It also aimed to be modern, with support for networked and multicore computing. Finally, working with Go is intended to be fast: it should take at most a few seconds to build a large executable on a single computer. To meet these goals required addressing a number of linguistic issues: an expressive but lightweight type system; concurrency and garbage collection; rigid dependency specification; and so on. These cannot be addressed well by libraries or tools; a new language was called for.*
>
> *The article Go at Google discusses the background and motivation behind the design of the Go language/*

Table 193: Data Sheet for Go.

Characteristic	Value
Paradigm	Multi-paradigm: concurrent, functional,[221] imperative, object-oriented [222][223]
Typing	Inferred, static, strong, structural[224][225]
Original author(s)	Robert Griesemer, Rob Pike, Ken Thompson
Developer	The Go Authors[226]
Initial release	10 November 2009

Characteristic	Value
Stable release	1.12.5 / May 6, 2019;[227]
Repository	https://git.kernel.org/pub/scm/git/git.git/
Written in	Go, assembly language (gc); C++ (gccgo)
Operating system	DragonFly BSD, FreeBSD, Linux, macOS, NetBSD, OpenBSD,[228] Plan 9,[229] Solaris, Windows
Available in	English
Type	Version control
License	BSD-style[230] + patent grant[231]
Website	https://golang.org/

221)
"First-Class Functions in Go". Retrieved November 14, 2018. Go supports ... a functional programming style in a strongly typed language.
222)
Is Go an object-oriented language?". Retrieved April 13, 2019. Although Go has types and methods and allows an object-oriented style of programming, there is no type hierarchy.
223)
"Go: code that grows with grace". Retrieved June 24, 2018. Go is Object Oriented, but not in the usual way.
224)
"Why doesn't Go have "implements" declarations?". golang.org. Retrieved October 1, 2015.
225)
Pike, Rob (December 22, 2014). "Rob Pike on Twitter". Retrieved March 13, 2016. Go has structural typing, not duck typing. Full interface satisfaction is checked and required.
226)
Text file LICENSE". The Go Programming Language. Google. Retrieved October 5, 2012.
227)
"Release History - The Go Programming Language". Retrieved April 19, 2019.
228)
"lang/go: go-1.4 – Go programming language". OpenBSD ports. December 23, 2014. Retrieved January 19, 2015.
229)
"Go Porting Efforts". Go Language Resources. cat-v. January 12, 2010. Retrieved January 18, 2010.
230)
Text file LICENSE". The Go Programming Language. Google. Retrieved October 5, 2012.
231)
Additional IP Rights Grant". The Go Programming Language. Google. Retrieved October 5, 2012.

Google: gRPC

return to Google page

Table 194: Data Sheet for gRPC

Characteristic	Value
Original author(s)	Google
Developer(s)	Google
Initial release	February 2015[232)]
Stable release	1.21.1 https://packages.grpc.io/
Repository	https://github.com/grpc/grpc/releases
Written in	
Operating system	Linux, Windows 7+,Mac[233)]
Software Languages	C/C++, C#, Dart, Go, Java, Node.js, PHP, Python, Ruby[234)]
Available in	English
Type	
License	Apache 2.0 [235)]
Website	https://grpc.io/about/

Overview

Source: The following is from the about gRPC page

gRPC is a modern open source high performance RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

The main usage scenarios:

- *Efficiently connecting polyglot services in microservices style architecture*
- *Connecting mobile devices, browser clients to backend services*
- *Generating efficient client libraries*

Core Features:

- *Idiomatic client libraries in 10 languages*
- *Highly efficient on wire and with a simple service definition framework*
- *Bi-directional streaming with http/2 based transport*
- *Pluggable auth, tracing, load balancing and health checking*

Architecture

Source: The following is from "Introduction to gRPC"

gRPC has two parts, the gRPC protocol, and the data serialization. By default gRPC utilizes Protobuf for serialization, but it is pluggable with any form of serialization you wish to use, with some caveats, which I will get to later.

The Protocol

The protocol itself is based on http2, and exploits many of its benefits. Google's development of SPDY laid down the first designs for what eventually became http2.

gRPC supports several built in features inherited from http2, such as compressing headers, persistent single TCP connections, cancellation and timeout contracts between client and server.

The protocol has built in flow control from http2 on data frames. This is very handy for insuring clients respect the throughput of your system, but does add an extra level of complexity when diagnosing issues in your infrastructure, because either client or server can set their own flow control values.

Load balancing (LB) is normally performed by the client, which chooses the server for a given request from a list provided by a Load Balancing server. The LB server will monitor the health of endpoints and use this and other factors to manage the list provided to clients. Clients will use a simple algorithm such as round-robin internally, but note that the LB server may apply more complex logic when compiling the list for a given client.

RPC Types

gRPC offers two essential types for client server communication.

Unary

Essentially these are synchronous requests made to the gRPC server with a single request that blocks until a response is received.

Streaming

Streaming is really powerful and can be accomplished in three different configurations: client pushing messages to a stream; server pushing messages to a stream; or bidirectional, where client and server are both sending data in two streams in the same method. In all cases the client initiates the RPC method.

Streams don't provide any acknowledgement of receipt until the stream completes, which can add complexity when the system needs to cope with node failures or network partitions. This can be mitigated by using a bidirectional stream to return ACKs. If a server is given a chance to kill a connection

gracefully a message will be returned indicating the last received message.

232)

Jason Smith, Container Solutions, Mar 1, 2017, https://container-solutions.com/introduction-to-grpc/

233), 234)

The gRPC about page https://grpc.io/about/

235)

gRPC FAQ, https://grpc.io/faq/

Google: Protocol Buffers

return to Google page

Source: The following is from Google "Protocol Buffers " site

> *Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler. You define how you want your data to be structured once, then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages.*

Table 195: Data Sheet for Protocol Buffers

Characteristic	Value
Developer	Google
Initial release	10 November 2009[236)]
Stable release	3.7.1 / March 26, 2019[237)]
Repository	https://github.com/protocolbuffers/protobuf
Written in	Go, assembly language (gc); C++ (gccgo)
Operating system	Any
Available in	English
Type	Version control
License	BSD
Website	https://developers.google.com/protocol-buffers/

236)
"Frequently Asked Questions | Protocol Buffers". Google Developers. Retrieved 2 October 2016.

237)
"Releases - google/protobuf". Retrieved 2 April 2019 – via GitHub.

IOTA

return to the de facto Standards area

Source: What is IOTA

> *IOTA is a revolutionary new transaction settlement and data transfer layer for the Internet of Things (IOT). It is based on a new distributed ledger technology, the Tangle, which overcomes the inefficiencies of current Blockchain designs and introduces a new way of reaching consensus in a decentralized Peer to Peer (P2P) system. Using IOTA, for the first time ever, people and machines can transfer money and/or data without any transaction fees in a trustless, permissionless, and decentralized environment. This means that even nano-payments are possible without the need for a trusted intermediary of any kind.*
>
> *IOTA is the missing puzzle piece for the Machine Economy to fully emerge and reach its true potential. IOTA is envisaged to be the public and permissionless backbone protocol for the IoT that enables true interoperability between all devices.*

Kinds of Nodes

Source: What is the difference between a Light Node and Full Node?

> *The IOTA GUI makes it possible to choose between a Full Node and Light Node. The Full Node automatically runs an IRI instance in the background, which in turn means that you need neighbors in order to participate in the Peer-to-Peer network to synchronize with the Tangle. If you run the Full Node, you have no “trust requirements”, as you are completely and independently participating in the network.*
>
> *The Light Node makes it possible to connect to a remote Full Node (whether it is your own Full Node, or someone else's publicly provided), in order to get the latest state of the network, most importantly fetching two transactions for validation which are required for issuing a transaction. When running a Light Node, it should be noted that your seed never leaves your wallet, you are still required to perform the Proof-of-Work, and all the sensitive work (such as signing) is done client-side. It also should be noted that the public Full Node providers are often overwhelmed with Light Node requests, so in order to reap the full benefits of the IOTA feeless and fast transaction settlement and data integrity technology, it is recommended to run a Full Node and not have to rely on the public Full Node providers (think of it like hitchhiking vs. having your own car).*

The IOTA Foundation

Source: What is the IOTA Foundation

The IOTA Foundation is the non-profit, open-source driven organization behind the IOTA Distributed Ledger Technology (DLT) and other related technologies. The IOTA Foundation houses the academic researchers, developers, industry experts, and other engineering and business professionals working on the development and adoption of the IOTA protocol for the machine economy and the IoT industry. Apart from research and development, the Foundation also fosters the adoption of its technologies into real-world production use-cases at scale by providing an open, collaborative ecosystem where companies, startups and developers alike can innovate with the technologies and build proof-of-concepts, case studies and pilots together. The IOTA Foundation is organized into five divisions:

- ***General administration**: Operations, communications, finance, human resources, and legal.*
- ***Social & Public Policy**: Social initiatives, collaborations with Governments and NGOs, and governmental regulations advocacy.*
- ***Research & Development**: Academic research, IOTA core protocol development, theory to working code, protocol standardization, and new technologies.*
- ***Ecosystem**: Community development, developer resources and advocacy, educational material, events and hackathons.*
- ***Business & Industry Streams**: Industry adoption, key account management, proof-of-concept development, strategic development, alliance management.*

Linux Foundation

return to the de facto Standards area

Create a New Page for Linux (e.g. *Linux Kubernetes*) **Full Name →**		Add page

Source: The following is from the "about" page of the Linux Foundation

> *The Linux Foundation has taken its experience and expertise supporting the Linux community to help establish, build, and sustain some of the most critical open source technologies. Its work today extends far beyond Linux, fostering innovation in every layer of the software stack. The Linux Foundation hosts projects spanning enterprise IT, embedded systems, consumer electronics, cloud, networking, and more.*
>
> *A few of these high-velocity projects that are helping redefine what's possible include Hyperledger for cross-industry blockchain technologies; Automotive Grade Linux, the open software platform for automotive applications; the Open Network Automation Platform project (ONAP) for real-time, policy-driven software automation of virtual network functions; and Kubernetes, the Cloud Native Computing Foundation project for production-grade container orchestration.*

de facto Standards

- Linux Foundation: Hyperledger
- Linux Foundation: OpenJS Foundation
- Kubernetes
- Node.js
- Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)
- Linux Foundation: Open Messaging
- ISO/IEC The Linux Standard Base 5 Specification Series (LSB 5)

Tools

The following are community tools that are part of the Linux Foundation.

- ***dep-checker*** *– A free dependency checker tool from The Linux Foundation, dep-checker performs a complete analysis of linkages between code packages.*
  http://git.linuxfoundation.org/dep-checker.git/
- ***FOSSology*** *– A Linux Foundation project, FOSSology is an open source license compliance software toolkit which can run license, copyright and export control scans from the command line. A database and web UI are also available to create compliance workflows.*
  https://www.fossology.org/
- ***janitor.git*** *– Code Janitor is an open source tool that helps evaluate source code for compliance*

with open source licenses. From The Linux Foundation, Code Janitor can be used with other products to check code. http://git.linuxfoundation.org/janitor.git/

- ***SPDX*** *– The Software Package Data Exchange (SPDX) specification is a standard format used to describe the components, licenses and copyrights associated with software packages. The SPDX standard aids compliance with free and open source software licenses by standardizing the way license information is shared between developers and companies. The SPDX specification is developed by the SPDX workgroup, which is hosted by The Linux Foundation. The group offers open source tools to help users of SPDX documents. https://spdx.org/tools*
- ***CII Best Practices Badging*** *– From The Linux Foundation, the Core Infrastructure Initiative (CII) Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify for free by using this web application to explain how they follow each best practice.https://bestpractices.coreinfrastructure.org/*

Linux Foundation: Hyperledger

return to the Linux Foundation

Source: About Linux Foundation Hyperledger

> ***Hyperledger*** *is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing and Technology.*
>
> *Not since the Web itself has a technology promised broader and more fundamental revolution than blockchain technology. A blockchain is a peer-to-peer distributed ledger forged by consensus, combined with a system for "smart contracts" and other assistive technologies. Together these can be used to build a new generation of transactional applications that establishes trust, accountability and transparency at their core, while streamlining business processes and legal constraints.*
>
> *Think of it as an operating system for marketplaces, data-sharing networks, micro-currencies, and decentralized digital communities. It has the potential to vastly reduce the cost and complexity of getting things done in the real world.*
>
> *Only an Open Source, collaborative software development approach can ensure the transparency, longevity, interoperability and support required to bring blockchain technologies forward to mainstream commercial adoption. That is what Hyperledger is about – communities of software developers building blockchain frameworks and platforms.*
>
> *Hyperledger launched in 2016 with a technical and organizational governance structure and 30 founding corporate members. Initially, the Hyperledger Technical Steering Committee welcomed two business blockchain framework codebases into incubation: Hyperledger Fabric, a codebase combining work by Digital Asset, libconsensus from Blockstream and OpenBlockchain from IBM; and Hyperledger Sawtooth, developed at Intel's incubation group.*

Linux Foundation: OpenJS Foundation

return to Linux Foundation

About

Source: About OpenJS Foundation

*Developers rely on a growing portfolio of open source technologies to create, test and deploy critical applications. By creating a center of gravity for the open source JavaScript ecosystem, the **OpenJS Foundation's** mission is to drive broad adoption and ongoing development of key JavaScript solutions and related technologies.*

The primary goals of the OpenJS Foundation are:

- *To promote the widespread adoption and continued development of key JavaScript and web solutions and related technologies.*
- *To facilitate collaboration within the JavaScript development community.*
- *To create a center of gravity for open source projects throughout the end-to-end JavaScript ecosystem guiding them toward open governance and diverse collaborator bases.*
- *To host the infrastructure to support hosted JavaScript open source projects.*
- *To enable, through advancement of Projects and strategic partnerships, an open and accessible web.*

Hosted Projects

Source: OpenJS Foundation Hosted Projects

We strongly believe in sharing best practices and reducing redundant administrative work across projects, particularly when it comes to non-technical governance. To that end we have established a Cross Project Council, or CPC, to centralize coordination among projects as well as certain technical governance and moderation processes. One of the CPC's primary functions will be to oversee the progression of projects between stages of their lifecycles.

Projects hosted by the OpenJS Foundation fall into one of four categories:

- ***Impact** stage is generally for large, mature projects.*
- ***Growth** stage is for projects which are actively mentored, and which intend to graduate to Impact stage.*
- ***At-Large** stage is for new projects, stable projects with minimal needs, and everything in between.*
- ***Incubation** stage is for projects which are in the process of joining the OpenJS Foundation.*
- ***Emeritus** stage is for projects which have completed their lifecycle and are retired.*

*In addition, the CPC oversees an **Incubation** process for projects which seek to be hosted by the OpenJS Foundation.*

The project lifecycle is documented in the Project Progression proposal.

Kubernetes

return to the Linux Foundation

Note: The following is an excerpt from the official Kubernetes (K8) home page. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 196: Data sheet for Linux Kubernetes

Title	Kubernetes
Acronym	K8s
Version	v1.18
Operating Systems	Darwin, Linux, Windows
Downloads	https://kubernetes.io/docs/setup/release/notes/#downloads-for-v1180
Repository	https://github.com/kubernetes/kubernetes
Supported Languages	
License	Apache License 2.0
Reference	https://kubernetes.io/docs/reference/

Kubernetes (K8s) *is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community*

It is based on the principles of:

- ***Planet Scale****: Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.*
- ***Never Outgrow****: Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.*
- ***Run Anywhere****: Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.*

Kubernetes features:

- **Service discovery and load balancing**: no need to use an unfamiliar service discovery mechanism.
- **Storage Orchestration**: ability to automatically mount storage mechanisms of choice.
- **Automated rollouts and rollbacks**: progressive change rollouts including application health monitoring.
- **Automatic bin packing**: automatically places containers based on resource requirements and other constraints.
- **Batch execution**: manages batch and CI workloads.
- **Service Topology**: Route service traffic based on cluster topology.
- **Self-healing**: manage failed and failing containers and dead nodes.

- **Secret and configuration management**: Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.
- **IPv4/IPv6 dual-stack**: Allocate of IPv4 and IPv6 addresses to Pods and Services
- **Horizontal scaling**: Scale applications with simple CLI and/or UI or in automatically in response to CPU usage.

Node.js

return to the Linux Foundation

> **Note:** The following is an excerpt from the official About Node.js site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 197: Data sheet for Node.js

Title	Node.js
Acronym	
Version	14.3.0
Operating Systems	Linux, macOS, Windows, SmartOS, FreeBSD, OpenBSD, IBM AIX
Downloads	https://nodejs.org/en/download/
Repository	https://github.com/nodejs/node
Supported Languages	JavaScript
License	MIT
Reference	https://nodejs.org/en/

About

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through callbacks at the beginning of a script, and at the end a server is started through a blocking call like `EventMachine::run()`. In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.

Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our `child_process.fork()`[238] API, and are designed to be easy to communicate with. Built upon that same interface is the `cluster`[239] module, which allows you to share sockets between processes to enable load balancing over your cores.

238)

https://nodejs.org/api/child_process.html#child_process_child_process_fork_modulepath_args_options

239)

https://nodejs.org/api/cluster.html

Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)

return to the Linux Foundation

> **Note**: The following is an excerpt from the official OpenMAMA site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

> ***OpenMAMA**, the Open Middleware Agnostic Messaging API, is a lightweight vendor-neutral integration layer for systems built on top of a variety of message orientated middlewares. The objective of OpenMAMA is to enable users to develop high-performance, event driven applications against a single standard API, while providing a mechanism for easily switching between middlewares as requirements evolve.*

Table 198: Data sheet for Open Middleware Agnostic Messaging API,

Title	Open Middleware Agnostic Messaging API,
Acronym	OpenMAMA
Version	6.2.3
Operating Systems	Linux Windows [240)]
Downloads	https://www.openmama.org/downloads
Repository	https://github.com/OpenMAMA/OpenMAMA/
Supported Languages	C, C++, C#, JNI[241)]
License	LGPL v2.1 [242)]
Reference	https://www.openmama.org/

240) , 241)
OpenMAMA Developer's Guide, 23 November 2012, https://www.openmama.org/sites/default/files/OpenMAMA%20Developers%20Guide%20JNI.pdf
242)
https://www.openmama.org/what-is-openmama/introduction-to-openmama

Linux Foundation: Open Messaging

return to the Linux Foundation

> **Note**: The following is an excerpt from the official Open Messaging site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

> *OpenMessaging, a vendor-neutral open standard for distributed messaging and streaming, which includes the establishment of industry guidelines and messaging, streaming specifications to provide a common framework for finance, e-commerce, IoT and big-data area. The design principles are the cloud-oriented, simplicity, flexibility, and language independent in distributed heterogeneous environments. Conformance to these specifications will make it possible to develop a heterogeneous messaging applications across all major platforms and operating systems.*

Table 199: Data sheet for Open Messaging,

Title	OpenMessaging
Acronym	
Version	under development
Operating Systems	Where ever Java SE runs
Software Language	Java[243)]
Cloud platforms	Alibaba Cloud, Azure, Google, and AWS) through OpenMessaging[244)]
Repository	https://github.com/openmessaging
Supported Languages	
License	Apache 2.0[245)]
Reference	http://openmessaging.cloud/

243) http://openmessaging.cloud/openmessaging-java/

244) https://github.com/cncf/toc/issues/103

245) https://github.com/cncf/toc/issues/103

ISO/IEC The Linux Standard Base 5 Specification Series (LSB 5)

return to the ISO Standards **OR** return to the Linux Foundation

Note: The LSB 5.0 Core specification set is an evolution of the ISO/IEC International Standard 23360, which corresponded to LSB 3.1. This edition is not to be considered an ISO standard.

Table 200: Data sheet for The Linux Standard Base 5 Specification Series

Title	The Linux Standard Base 5 Specification Series
Acronym	LSB
Version	2015
Document Number	LSB Common 5.0 Edition
Release Date	23 Jun 2015
Reference	http://refspecs.linuxbase.org/LSB_5.0.0/LSB-Common/LSB-Common/book1.html

Note: The following is an excerpt from the official Linux Foundation LSB documentation. It is provided here as a connivence and is not authoritative. Refer to the original document as the authoritative reference.

General

The Linux Standard Base (LSB) defines a system interface for compiled applications and a minimal environment for support of installation scripts. Its purpose is to enable a uniform industry standard environment for high-volume applications conforming to the LSB.

The LSB specification set is divided into modules, each of which provides fundamental system interfaces, libraries, and runtime environment upon which all conforming applications and libraries using that module depend.

The modules of the Linux Standard Base are:

- ***LSB Core** - core components*
- ***LSB Desktop** - desktop related components*
- ***LSB Languages** - runtime languages*
- ***LSB Imaging** - printing and scanning*
- ***LSB Trial Use** - components that are not yet mandatory*

Interfaces described in the LSB Core module specification are supplemented by other LSB module specifications. All other modules depend on the presence of LSB Core.

These specifications are composed of two basic parts: a common part describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part describing the parts of the interface that vary by processor architecture. Together, the common part and the relevant architecture-specific part for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture. Whenever a section of the common part is supplemented by architecture-specific information, the

common part includes a reference to the architecture-specific part. Architecture-specific parts of of an LSB module specification may also contain additional information that is not referenced in the common part.

The LSB contains both a set of Application Program Interfaces (APIs) and Application Binary Interfaces (ABIs). APIs may appear in the source code of portable applications, while the compiled binary of that application may use the larger set of ABIs. A conforming implementation provides all of the ABIs listed here. The compilation system may replace (e.g. by macro definition) certain APIs with calls to one or more of the underlying binary interfaces, and may insert calls to binary interfaces as needed.

The LSB is primarily a binary interface definition. Not all of the source level APIs available to applications may be contained in this specification.

Microsoft

return to the de facto Standards area

Create a New Page for Microsoft (e.g. *Windows 10*) **Full Name →** Add page

de facto Standards

- Microsoft: Windows API

Microsoft: Windows API

return to Microsoft

Source: Windows API

The Windows API, informally WinAPI, is Microsoft's core set of application programming interfaces (APIs) available in the Microsoft Windows operating systems. The name Windows API collectively refers to a number of different platform implementations that are often referred to by their own names (for example, Win32 API); see the versions section. Almost all Windows programs interact with the Windows API; on the Windows NT line of operating systems, a small number (such as programs started early in the Windows startup process) use the Native API.

Overview

The functionality provided by the Windows API can be grouped into eight categories:

Base Services

Provide access to the fundamental resources available to a Windows system. Included are things like file systems, devices, processes, threads, and error handling. These functions reside in kernel.exe, krnl286.exe or krnl386.exe files on 16-bit Windows, and kernel32.dll on 32-bit Windows.

Advanced Services

Provide access to functionality additional to the kernel. Included are things like the Windows registry, shutdown/restart the system (or abort), start/stop/create a Windows service, manage user accounts. These functions reside in advapi32.dll on 32-bit Windows.

Graphics Device Interface

Provides functionality for outputting graphical content to monitors, printers and other output devices. It resides in gdi.exe on 16-bit Windows, and gdi32.dll on 32-bit Windows in user-mode. Kernel-mode GDI support is provided by win32k.sys which communicates directly with the graphics driver.

User Interface

Provides the functionality to create and manage screen windows and most basic controls, such as buttons and scrollbars, receive mouse and keyboard input, and other functionality associated with the Graphical User Interface (GUI) part of Windows. This functional unit resides in user.exe on 16-bit Windows, and user32.dll on 32-bit Windows. Since Windows XP versions, the basic controls reside in comctl32.dll, together with the common controls (Common Control Library).

Common Dialog Box Library

Provides applications the standard dialog boxes for opening and saving files, choosing color and font, etc. The library resides in a file called commdlg.dll on 16-bit Windows, and comdlg32.dll on 32-bit Windows. It is grouped under the User Interface category of the API.

Common Control Library

Gives applications access to some advanced controls provided by the operating system. These include things like status bars, progress bars, toolbars and tabs. The library resides in a DLLfile called commctrl.dll on 16-bit Windows, and comctl32.dll on 32-bit Windows. It is grouped under the User Interface category of the API.

Windows Shell

Component of the Windows API allows applications to access the functionality provided by the operating system shell, as well as change and enhance it. The component resides in shell.dll on 16-bit Windows, and shell32.dll on 32-bit Windows. The Shell Lightweight Utility Functions are in shlwapi.dll. It is grouped under the User Interface category of the API.

Network Services

Give access to the various networking capabilities of the operating system. Its sub-components include NetBIOS, Winsock, NetDDE, RPC and many others. This component resides in netapi32.dll on 32-bit Windows.

Versions

Win16 - 16 bit version
Win32 - 32 bit version

Win64 - 64 bit version
WinCE - Embedded Compact version

Oracle

return to the de facto Standards area

Create a New Page for Oracle (e.g. *Java® Language Specification SE 8 Edition*) **Full Name →**	Add page

Oracle Corporation is an American multinational computer technology corporation headquartered in Redwood Shores, California. The company sells database software and technology, cloud engineered systems, and enterprise software products—particularly its own brands of database management systems. In 2018, Oracle was the third-largest software company by revenue.
https://en.wikipedia.org/wiki/Oracle_Corporation

de facto Standards

- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API

Oracle: The Java® Language Specification SE 8 Edition

return to Oracle de facto Standards

Table 201: Data sheet for The Java® Language Specification

Title	The Java® Language Specification
Acronym	Java
Version	SE 8 Edition
Document Number	
Release Date	13 February 2015
Reference	https://docs.oracle.com/javase/specs/jls/se8/html/

Note: The following is an excerpt from the official site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Introduction

The Java® programming language is a general-purpose, concurrent, class-based, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language. The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included. It is intended to be a production language, not a research language, and so, as C. A. R. Hoare suggested in his classic paper on language design, the design has avoided including new and untested features.

The Java programming language is strongly and statically typed. This specification clearly distinguishes between the compile-time errors that can and must be detected at compile time, and those that occur at run time. Compile time normally consists of translating programs into a machine-independent byte code representation. Run-time activities include loading and linking of the classes needed to execute a program, optional machine code generation and dynamic optimization of the program, and actual program execution.

The Java programming language is a relatively high-level language, in that details of the machine representation are not available through the language. It includes automatic storage management, typically using a garbage collector, to avoid the safety problems of explicit deallocation (as in C's free or C++'s delete). High-performance garbage-collected implementations can have bounded pauses to support systems programming and real-time applications. The language does not include any unsafe constructs, such as array accesses without index checking, since such unsafe constructs would cause a program to behave in an unspecified way.

The Java programming language is normally compiled to the bytecode instruction set and binary format defined in The Java Virtual Machine Specification, Java SE 8 Edition.

Oracle: The Java® Virtual Machine Specification JVM

return to Oracle de facto Standards

Table 202: Data sheet for The Java® Virtual Machine Specification

Title	The Java® Virtual Machine Specification
Acronym	JVM
Version	Java SE 8 Edition
Document Number	
Release Date	13 February 2013
Reference	https://docs.oracle.com/javase/specs/jvms/se8/html/

Note: The following is an excerpt from the official site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

The Java Virtual Machine

The Java Virtual Machine is the cornerstone of the Java platform. It is the component of the technology responsible for its hardware- and operating system-independence, the small size of its compiled code, and its ability to protect users from malicious programs.

The Java Virtual Machine is an abstract computing machine. Like a real computing machine, it has an instruction set and manipulates various memory areas at run time. It is reasonably common to implement a programming language using a virtual machine; the best-known virtual machine may be the P-Code machine of UCSD Pascal.

The first prototype implementation of the Java Virtual Machine, done at Sun Microsystems, Inc., emulated the Java Virtual Machine instruction set in software hosted by a handheld device that resembled a contemporary Personal Digital Assistant (PDA). Oracle's current implementations emulate the Java Virtual Machine on mobile, desktop and server devices, but the Java Virtual Machine does not assume any particular implementation technology, host hardware, or host operating system. It is not inherently interpreted, but can just as well be implemented by compiling its instruction set to that of a silicon Central Processing Unit (CPU). It may also be implemented in microcode or directly in silicon.

The Java Virtual Machine knows nothing of the Java programming language, only of a particular binary format, the class file format. A class file contains Java Virtual Machine instructions (or bytecodes) and a symbol table, as well as other ancillary information.

For the sake of security, the Java Virtual Machine imposes strong syntactic and structural constraints on the code in a class file. However, any language with functionality that can be expressed in terms of a valid class file can be hosted by the Java Virtual Machine. Attracted by a

generally available, machine-independent platform, implementors of other languages can turn to the Java Virtual Machine (VM) as a delivery vehicle for their languages.

The Java Virtual Machine specified here is compatible with the Java SE 8 platform, and supports the Java programming language specified in The Java Language Specification, Java SE 8 Edition.

Oracle: Java logger API

return to Oracle de facto Standards

> **Note**: The following is an excerpt from the official TTTTT site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 203: Data sheet for Java logger API

Title	Java logger API (Java Platform SE-8)
Acronym	Logger
Version	Java logger API
Operating Systems	Java VM
Downloads	https://www.oracle.com/technetwork/java/javase/downloads/index.html
Supported Languages	
License	Oracle Proprietary
Reference	https://docs.oracle.com/en/java/javase/12/docs/api/java.logging/module-summary.html

Overview

Provides the classes and interfaces of the Java™ 2 platform's core logging facilities. The central goal of the logging APIs is to support maintaining and servicing software at customer sites. There are four main target uses of the logs:

1. *Problem diagnosis by end users and system administrators. This consists of simple logging of common problems that can be fixed or tracked locally, such as running out of resources, security failures, and simple configuration errors.*
2. *Problem diagnosis by field service engineers. The logging information used by field service engineers may be considerably more complex and verbose than that required by system administrators. Typically such information will require extra logging within particular subsystems.*
3. *Problem diagnosis by the development organization. When a problem occurs in the field, it may be necessary to return the captured logging information to the original development team for diagnosis. This logging information may be extremely detailed and fairly inscrutable. Such information might include detailed tracing on the internal execution of particular subsystems.*
4. *Problem diagnosis by developers. The Logging APIs may also be used to help debug an application under development. This may include logging information generated by the target application as well as logging information generated by lower-level libraries. Note however that while this use is perfectly reasonable, the logging APIs are not intended to replace the normal debugging and profiling tools that may already exist in the development environment.*

The key elements of this package include:

- ***Logger** The main entity on which applications make logging calls. A Logger object is used to log messages for a specific system or application component.*

- ***LogRecord*** *Used to pass logging requests between the logging framework and individual log handlers.*
- ***Handler*** *Exports LogRecord objects to a variety of destinations including memory, output streams, consoles, files, and sockets. A variety of Handler subclasses exist for this purpose. Additional Handlers may be developed by third parties and delivered on top of the core platform.*
- ***Level*** *Defines a set of standard logging levels that can be used to control logging output. Programs can be configured to output logging for some levels while ignoring output for others. Filter: Provides fine-grained control over what gets logged, beyond the control provided by log levels. The logging APIs support a general-purpose filter mechanism that allows application code to attach arbitrary filters to control logging output.*
- ***Formatter*** *Provides support for formatting LogRecord objects. This package includes two formatters, SimpleFormatter and XMLFormatter, for formatting log records in plain text or XML respectively. As with Handlers, additional Formatters may be developed by third parties.*

The Logging APIs offer both static and dynamic configuration control. Static control enables field service staff to set up a particular configuration and then re-launch the application with the new logging settings. Dynamic control allows for updates to the logging configuration within a currently running program. The APIs also allow for logging to be enabled or disabled for different functional areas of the system. For example, a field service engineer might be interested in tracing all AWT events, but might have no interest in socket events or memory management.

Talk Openly Develop Openly (TODO)

return to the de facto Standards area

Create a New Page for TODO **Full Name** →		Add page

Source: The following is from the TODO About page

> *Open source is part of the fabric of each of our companies. Between us, our open source programs enable us to use, contribute to, and maintain, thousands of projects - both large and small.*
>
> *These programs face many challenges, such as ensuring high-quality and frequent releases, engaging with developer communities, and contributing back to other projects effectively.*
>
> *The members of this group are committed to working together in order to overcome these challenges. We will be sharing experiences, developing best practices, and working on common tooling. But we can't do this alone. If you are a company using or sharing open source, we welcome you to join us and help make this happen.*
>
> ***TODO**: **talk openly, develop openly**. We believe we will improve our open source programs - and our contributions to the open source movement as a whole - by working together.*

de facto Standards

Build an Open Source Program

- TODO: How to create an open source program
- TODO: Measuring your open source program's success
- TODO: Tools for managing open source programs

Manage an Open Source Program

There have been many books written on this subject: [246)]

- TODO: Using open source code
- TODO: Participating in open source communities
- TODO: Recruiting open source developers
- TODO: Starting an open source project
- TODO: Improve your open source development impact
- TODO: Shutting down an open source project
- TODO: Building leadership in an open source community
- TODO: Setting an Open Source Strategy

246)

TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

TODO: How to create an open source program

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "How to create an open source program"

A central open source program office is a designated place where open source is supported, nurtured, shared, explained, and grown inside a company. With such an office in place, businesses can establish and execute on their open source strategies in clear terms, giving their leaders, developers, marketers, and other staff the tools they need to make open source a success within their operations.

This guide aims to help you figure out why and how to establish a program to manage the open source use and creation inside your company, as well as to show how your developers can make their own contributions to open source projects outside your operations. It will cover topics for open source offices including: roles and responsibilities, corporate structures, elements of an open source management program, how to choose and hire an open source program manager, and more.

Contents

TODO: Measuring your open source program's success

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Measuring your open source program's success"

Open source program managers must demonstrate the ROI of their efforts. This guide provides an overview of some of the standard ways that organizations evaluate their open source programs, projects, and contributions.

Learn what to measure, how to define success, and how to best use this information to advance your open source program objectives, demonstrate effectiveness, and gain support.

Contents

TODO: Tools for managing open source programs

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Tools for managing open source programs"

The road to strategic use of open source starts with a carefully planned, organized, and empowered open source program office to guide and manage its creation, distribution, and use. But that's just a first step. To get such an office underway and running smoothly, you need the right tools. These mission-critical tools will be used to track goals and metrics in departments from engineering and legal to executive leadership, PR, and marketing, and give employees all the resources they need to gather data, provide snapshots of performance, and manage the daily use of open source within your company.

This guide provides details and scenarios for how to get your open source tool collection started, including information about the most important tools to use to track and manage your open source projects. Many of the tools have been created and open-sourced by The Linux Foundation and other leaders in the field, providing free and easy access for your projects. You'll also find an example dashboard setup which combines and displays information from multiple tools.

Contents

TODO: Using open source code

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Using open source code"

One of the most important responsibilities of an open source program office is ensuring that your organization meets its legal obligations when integrating open source code with proprietary and third-party source code in your commercial products.

You need to establish guidelines on how developers can use open source code, and detailed processes to track where open source code is coming from, how it's licensed, and where it ultimately ends up. This guide gets you started with a baseline compliance program for using, releasing, and distributing open source code.

Contents

TODO: Participating in open source communities

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Participating in open source communities"

Open source has become the de facto way to build software – not only in tech, but across diverse industries. As companies use open source code to build their own commercial products and services, they also see the strategic value of contributing back to those projects.

However, diving in without an understanding of those projects, their communities, and how they operate can lead to frustrations for those companies as well as the open source communities. Approaching open source contributions without a strategy can tarnish a company's reputation in the open source community and incur legal risks.

This guide covers what it means to contribute to open source as an organization and how to become a good corporate citizen. Learn how open source projects are structured, how to contribute, why it's important to devote internal developer resources to participation, and why it's important to create a strategy for open source participation and management.

Contents

TODO: Recruiting open source developers

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Recruiting open source developers"

Experienced open source developers are in short supply. To attract top talent, companies have to do more than hire a recruiter or place an ad on a popular job site.

Your open source program can become one of your most effective recruiting tools. This guide covers how organizations can recruit developers, or build internal talent, by building an open source culture, contributing to open source communities, and creating open source projects.

Contents

TODO: Starting an open source project

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Starting an open source project"

Once a company has participated in open source communities long enough to build a reputation, it's in a position to launch its own open source projects. It's at this stage of open source participation that companies can realize the greatest benefits from open collaboration. You can open source proprietary projects that could be of use to the community. Another common avenue is to create new open source projects from scratch and benefit from collaboration among external developers at the outset.

This guide was created to help enterprises already well versed in open source learn what they need to know to start their own open source projects. We'll take you through the process, from deciding on what to open source, to budget and legal considerations, and more. The road to creating an open source project may be foreign, but major enterprises including Google, IBM, Facebook, Twitter and Microsoft have blazed the trail for you. Follow this guide for topical and helpful advice and you will be on your way.

Contents

TODO: Improve your open source development impact

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Improve your open source development impact"

Open source development requires a different approach to software engineering than many organizations are accustomed to. It becomes easier if you have a clear plan to follow. Fortunately, many companies and individuals have already forged a path to success by contributing to significant open source projects in strategic ways. This practical guide will help you and your company improve your internal development process and prepare you to contribute to the open source projects that matter most to your company. Contributing to the Linux kernel is one of the hardest challenges for open source developers. So we'll use this case as an example for this guide. Fortunately, the guidance will apply to nearly any open source project you'll face.

Contents

TODO: Shutting down an open source project

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Shutting down an open source project"

This Open Source Guide is designed to offer advice about how your enterprise and your development team can plan for the day when you are ready to end or move away from an unneeded open source project. By shutting down the project gracefully or by transitioning it to others who can continue the work, your enterprise can responsibly oversee the life cycle of the effort. In this way, you can also set proper expectations for users, ensure that long-term project code dependencies are supported, and preserve your company's reputation within the open source community as a responsible participant.

This guide will help you decide when a project is no longer useful, understand how to disengage from a project, and determine what to do about its code, repositories, websites, wikis, and other project assets as you head in a new direction.

Contents

TODO: Building leadership in an open source community

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Building leadership in an open source community"

Integrating into open source communities takes time and effort and requires a new approach to product development. Where traditional, proprietary development requires secrecy and a management hierarchy, open source development requires openness and values consensus. Code contributions, not title or position, are what determine influence and technical direction in an open source project.

Open source projects are developed in diverse and geographically dispersed communities that have their own rules, conventions, tools, and processes. Simply put, each community has its own unique culture and it takes time to establish the trust, ways of collaborating, and cultural understanding required to be effective in open source.

This guide explains how organizations can build leadership and influence within the open source projects they're involved in and on which they are commercially dependent. Learn about leadership culture and roles within a project, how decisions are made, how an organization can build leadership, and tips for being a good leader in open source communities.

Contents

1. Why build leadership in an open source project

- *Leadership culture in open source*
- *Corporate vs open source leadership*
- *Governance overview*
- *Culture overview*

2. Leadership roles in a collaborative project

- *Leadership roles matrix*
- *How to become a leader*

TODO: Setting an Open Source Strategy

return to the TODO defacto Standards

Overview

There are no formal standards on Open Source projects. There are only guides.

Source: The following is from TODO on "Setting an Open Source Strategy"

The majority of companies that use open source do not necessarily understand the benefits to their organization and do not have a strategy aligned with their business needs. Furthermore, only about half of these companies report practicing basic open source management, such as community development, code maintenance, and the like, according to the latest Future of Open Source survey.

Creating and documenting an open source strategy is an essential first step to realizing ROI with open source as open source can have a positive or negative effect across several dimensions related to an organization: strategic/business, operational, technological and financial. Your open source strategy connects the plans for managing, participating in, and creating open source software with the business objectives that the plans serve. This can open up many opportunities and catalyze innovation.

Contents

GIT (Revision Control)

return to the de facto Standards area

Source: The following is from the English Wikipedia on GIT

> *Git (/gɪt/) [247] is a distributed version-control system for tracking changes in source code during software development.[248] It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed,[249] data integrity, [250] and support for distributed, non-linear workflows. [251]*
>
> *Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. [252] Its maintainer since 2005 is and continues to be Junio Hamano.*
>
> *As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.[253]*
>
> *Git is free and open-source software distributed under the terms of the GNU General Public License version 2.*

Table 204: Data Sheet for GIT

Characteristic	Value
Original author(s)	Linus Torvalds[254]
Developer(s)	Junio Hamano and others[255]
Initial release	7 April 2005
Stable release	2.21.0 / 24 February 2019 [256]
Repository	https://git.kernel.org/pub/scm/git/git.git/
Written in	C, Shell, Perl, Tcl, Python[257]
Operating system	POSIX: Linux, Windows, macOS
Available in	English
Type	Version control
License	
Website	https://git-scm.com/

247) “Tech Talk: Linus Torvalds on git (at 00:01:30)”. YouTube. Archived from the original on 20 December 2015. Retrieved 20 July 2014.

248) Anthony; Huff, Kathryn D. (2015). Effective Computation in Physics. O'Reilly Media, Inc. p. 351. ISBN 9781491901595. Archived from the original on 7 May 2016. Retrieved 20 April 2016.

249) Torvalds, Linus (7 April 2005). “Re: Kernel SCM saga.” linux-kernel (Mailing list). “So I'm writing some scripts to try to track things a whole lot faster.”

250)
Torvalds, Linus (10 June 2007). “Re: fatal: serious inflate inconsistency”. git (Mailing list).
251)
Linus Torvalds (3 May 2007). Google tech talk: Linus Torvalds on git. Event occurs at 02:30. Archived from the original on 28 May 2007. Retrieved 16 May 2007.
252)
“A Short History of Git”. Pro Git (2nd ed.). Apress. 2014. Archived from the original on 25 December 2015. Retrieved 26 December 2015.
253)
Chacon, Scott (24 December 2014). Pro Git (2nd ed.). New York, NY: Apress. pp. 29–30. ISBN 978-1-4842-0077-3. Archived from the original on 25 December 2015.
254)
“Initial revision of “git”, the information manager from hell”. Github. 8 April 2005. Archived from the original on 16 November 2015. Retrieved 20 December 2015.
255)
“Commit Graph”. Github. 8 June 2016. Archived from the original on 20 January 2016. Retrieved 19 December 2015.
256)
“Releases - git/git”. Archived from the original on 20 September 2017. Retrieved 25 February 2019.
257)
“Git Source Code Mirror”. Archived from the original on 8 February 2017. Retrieved 1 January 2017.

InterPlanetary File System (IPFS)

return to the de facto Standards area

Source: The following is from the white paper IPFS - Content Addressed, Versioned, P2P File System

Table 205: Data Sheet for InterPlanetary File System (IPFS).

Characteristic	Value
Original author(s)	Juan Benet and Protocol Labs
Developer(s)	Protocol Labs
Initial release	24 July 2014
Stable release	0.4.19 / 1 March 2019
Repository	https://github.com/ipfs/ipfs
Written in	Protocol implementations: Go (reference implementation), JavaScript, C,[1] Python Client libraries: Go, Java, JavaScript, Python, Scala, Haskell, Swift, Common Lisp, Rust, Ruby, PHP, C#, Erlang
Operating system	FreeBSD, Linux, macOS, Windows
Available in	English
Type	Protocol, distributed file system, content delivery network
License	MIT license
Website	https://ipfs.io/

Abstract

The InterPlanetary File System (IPFS) is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. In some ways, IPFS is similar to the Web, but IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. In other words, IPFS provides a high throughput content-addressed block storage model, with content addressed hyper links. This forms a generalized Merkle DAG, a data structure upon which one can build versioned file systems, blockchains, and even a Permanent Web. IPFS combines a distributed hashtable, an incentivized block exchange, and a self-certifying namespace. IPFS has no single point of failure, and nodes do not need to trust each other.[258)]

258)
"IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)", Juan Denet, 24 July 2014, https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf

Jenkins (Continuous Delivery)

return to the de facto Standards area

Source: The following is from the English Wikipedia on Jenkins(software)

> *Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, TD/OMS, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands. The creator of Jenkins is Kohsuke Kawaguchi.* [259] *Released under the MIT License, Jenkins is free software.* [260]
>
> *Builds can be triggered by various means, for example by commit in a version control system, by scheduling via a cron-like mechanism and by requesting a specific build URL. It can also be triggered after the other builds in the queue have completed. Jenkins functionality can be extended with plugins.*
>
> *The Jenkins project was originally named Hudson, and was renamed after a dispute with Oracle, which had forked the project and claimed rights to the project name. The Oracle fork, Hudson, continued to be developed for a time before being donated to the Eclipse Foundation. Oracle's Hudson is no longer maintained* [261] [262] *and was announced as obsolete in February 2017.* [263]

Table 206: Data Sheet for Jenkins.

Characteristic	Value
Original author(s)	Kohsuke Kawaguchi
Initial release	2 February 2011;
Stable release	2.164.1[2] / 13 March 2019;
Repository	github.com/jenkinsci/jenkins
Written in	Java
Platform	Java SE
Available in	English
Type	Continuous Delivery
License	MIT License
Website	https://jenkins.io/

259) Dan Dyer. “Why are you still not using Hudson?”. Retrieved 2008-05-21.

260) Kawaguchi, Kohsuke; et al. “Use Hudson: License”. Archived from the original on February 7, 2009. Retrieved January 30, 2011.

261) “About Jenkins”. Eclipse Wiki: Jenkins. Retrieved 6 August 2017.

262) “About Jenkins”. Wayback Machine: Eclipse Wiki, first available on 6 August 2017.Archive index at the

Wayback Machine
263)

“About Jenkins”. Eclipse Wiki history.

Jira (Bug tracking system)

return to the de facto Standards area

Source: The following is from the English Wikipedia on Jira

> *ira (/ˈdʒiːrə/ JEE-rə) [264] is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management. The product name is a truncation of Gojira, the Japanese word for Godzilla, [265] which is a reference to a competitor, Bugzilla.*

Table 207: Data Sheet for Jira.

Characteristic	Value
Developer(s)	Atlassian
Initial release	2002;
Stable release	8.1.0
Written in	Java
Operating system	Cross Plaform (Java SE)
Available in	English
Sownload	Jira Sofware Download
Type	Bug tracking system, project management software
License	**Proprietary**, free for use by official non-profit organizations, charities, and open-source projects, but not governmental, academic or religious organizations[266)][267)]
Website	https://www.atlassian.com/software/jira

264) "About us". Atlassian.com official website. Retrieved 27 February 2012.

265) "What does JIRA mean?". Atlassian.com official website. Retrieved 16 March 2012.

266) "Open Source Project License Request". Atlassian.com official website. Retrieved 9 November 2012.

267) "Community License Request". Atlassian.com official website. Retrieved 9 November 2012.

Participating in Open Source Communities

return to the de facto Standards area

Source: The following is from the Guide "Participating in Open Source Communities"

Open source has become the de facto way to build software – not only in tech, but across diverse industries. As companies use open source code to build their own commercial products and services, they also see the strategic value of contributing back to those projects.

However, diving in without an understanding of those projects, their communities, and how they operate can lead to frustrations for those companies as well as the open source communities. Approaching open source contributions without a strategy can tarnish a company's reputation in the open source community and incur legal risks.

This guide covers what it means to contribute to open source as an organization and how to become a good corporate citizen. Learn how open source projects are structured, how to contribute, why it's important to devote internal developer resources to participation, and why it's important to create a strategy for open source participation and management.

Contents

———-

NOTE: *These resources were created in partnership with the TODO (Talk Openly, Develop Openly) Group – the professional open source program networking group at The Linux Foundation. A special thanks goes out to the open source program managers who contributed their time and knowledge to making these comprehensive guides. Participating companies include Autodesk, Comcast, Dropbox, Facebook, Google, Intel, Microsoft, Netflix, Oath (Yahoo + AOL), Red Hat, Salesforce, Samsung and VMware. To learn more, visit: https://todogroup.org/ todogroup.org//*

ZeroMQ Distributed Messaging

return to the de facto Standards area

Source: ØMQ - The Guide

Table 208: Data Sheet for ZeroMQ (ØMQ).

Characteristic	Value
Original author(s)	Pieter Hintjens
Developer(s)	iMatrix
Initial release	2010
Stable release	4.3.1
API Documentation	http://api.zeromq.org/4-3:_start
Repository	http://zeromq.org/intro:get-the-software or https://github.com/zeromq/libzmq
Written in	Mainly in C, but also in PHP, Java, Python, Lua, and Haxe
Example Languages	C++, C#, CL, Delphi, Erlang, F#, Felix, Haskell, Objective-C, Ruby, Ada, Basic, Clojure, Go, Haxe, Node.js, ooc, Perl, and Scala
Operating system	runs on most operating systems
Guide	http://zguide.zeromq.org/page:all
Available in	English and 27 languages other language
Type	Messaging Service
License	LGHPLv3
Website	http://zeromq.org/

Abstract

ZeroMQ (also known as ØMQ, 0MQ, or zmq) looks like an embeddable networking library but acts like a concurrency framework. It gives you sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast. You can connect sockets N-to-N with patterns like fan-out, pub-sub, task distribution, and request-reply. It's fast enough to be the fabric for clustered products. Its asynchronous I/O model gives you scalable multicore applications, built as asynchronous message-processing tasks. It has a score of language APIs and runs on most operating systems. ZeroMQ is from iMatix and is LGPLv3 open source.

ZeroMQ Message Transport Protocol (ZMTP)

return to the de facto Standards area

Source: ØMQ - The Guide

Table 209: Data Sheet for ZeroMQ Message Transport Protocol (ZMTP)

Characteristic	Value
Original author(s)	Pieter Hintjens
Developer(s)	iMatrix
Initial release	v 3.0
Stable release	None
API Documentation	https://rfc.zeromq.org/spec:23/ZMTP/
Repository	https://github.com/zeromq/zmtp
Written in	C
Example Languages	C
Operating system	
Guide	https://rfc.zeromq.org/spec:23/ZMTP/
Available in	English
Type	Transport Protocol
License	LGHPLv3
Website	http://zeromq.org/

Abstract

The ZeroMQ Message Transport Protocol (ZMTP) is a transport layer protocol for exchanging messages between two peers over a connected transport layer such as TCP. This document describes ZMTP 3.0.

Goals

The ZeroMQ Message Transport Protocol (ZMTP) is a transport layer protocol for exchanging messages between two peers over a connected transport layer such as TCP. This document describes version 3.0 of ZMTP. ZMTP solves a number of problems we face when using TCP carry messages:

- *TCP carries a stream of octets with no delimiters, but we want to send and receive discrete messages. Thus, ZMTP reads and writes frames consisting of a size and a body.*
- *We need to carry metadata on each frame (such as, whether the frame is part of a multipart message, or not). ZMTP provides a flags field in each frame for metadata.*
- *We need to be able to talk to older implementations, so that our framing can evolve without breaking existing implementations. ZMTP defines a greeting that announces the implemented version number, and specifies a method for version negotiation.*

- *We need security so that peers can be sure of the identity of the peers they talk to, and so that messages cannot be tampered with, nor inspected, by third parties. ZMTP defines a security handshake that allows peers to create a secure connection.*
- *We need a range of security protocols, from clear text (no security, but fast) to fully authenticated and encrypted (secure, but slow). Further, we need the freedom to add new security protocols over time. ZMTP defines a way for peers to agree on an extensible security mechanism.*
- *We need a way to carry metadata about the connection, after the security handshake. ZMTP defines a standard set of metadata properties (socket type, identity, etc.) that peers exchange after the security mechanism.*
- *We need to write down these solutions in a way that is easy for teams to implement on any platform and in any language. ZMTP is thus specified as a formal protocol (this document) and made available to teams under a free license.*
- *We need guarantees that people will not create private forks of ZMTP, thus breaking interoperability. ZMTP is thus licensed under the GPLv3, so that any derived versions must also be made available to users of software that implements it.*

Appendix C: Tools

return to the Public Area **OR** return to Reference Architecture (RA)

As with all projects, DIDOs require tools to support their development, management, and operation. The following lists of tools are meant to be informative and by no means exhaustive.

Community Tools

The following tools are used to aid in the development of software, particularly the development of Open Source Software (OSS).

- Tools: Open Source Paradigm
- Tools: Source Code Scanning and License Compliance
- Tools: Bug and Issue Tracking
- Tools: Archiving and Release Management
- Tools: Tracking Project Health
- Tools: Code Reviews
- Tools: Contributor License Agreements (CLA)
- Tools: GitHub Management at Corporate Scale
- Tools: Project Quality
- Tools: Logging Tools

Network Traffic Analysis Tools

- Tools: Network Traffic Analysis

Tools for communications and collaboration

- Tools TBD

Tools for corporate-scale GitHub management

- Tools TBD

Tools: Open Source Paradigm

Return to Tools Area

Using a DIDO is not just a simple shift in policies, procedures and practices. It is a change in the architectural paradigm from centralized control to distributed requiring a change to how the system is normalized into systems, subsystems, components, etc. It also requires a shift in basic underlying principles of the system. DIDOs generally are:

- Comprised of thousands if not millions of independent nodes
- Outside the control of any one individual or corporation
- Lacking centralized authority with decisions being made by consensus

The DIDO architecture does not represent a single unified enterprise but rather it represents a confederation of domains loosely defined that requires systems integration (SI)[268]. Although SI is not new to enterprises, the granularity and kinds of the components requires a rethink. Within the DIDO environment, the definition of a platform shifts from hardware, operating system, software languages, and services (i.e., web, app, database, etc.) components to the DIDO Platform components. It is the responsibility of the DIDO Platform to isolate the enterprise from the traditional platform concerns.

The granularity of the data elements within an enterprise can also shift to smaller more isolated objects that represent only a portion of the traditional Data Model (DM). In other words, the enterprise's data model is not going to be deployed into a single DIDO and nor should it. The Enterprise data stores will continue and will be augmented by the DIDO. Some data will reside completely within the Enterprise data stores, some data will reside completely within the DIDO and some data will straddle both. Data that straddles both needs more procedures and policies defined to ensure data integrity.

Relevant Open Source Standards

The cultural shift from a stove-piped corporate or enterprise culture with almost complete control to being a systems integrator participating in numerous distributed communities that cover a wide range of domains requires a comiited leadership and a concerted effort by all the players.

Technical Standards

- None at this time.

de facto Standards

- There are none at this time, but there are guides on participating in Open Source initiatives. There are many written on this subject. **Talk Openly Develop Openly** (TODO) provides an extensive reading list.[269]. TODO also provides the following excellent guide as a place to start.
- TODO: Participating in open source communities

268)

System Integrator - An individual or organization that builds systems from a variety of diverse components. With increasing complexity of technology, more customers want complete solutions to information problems, requiring hardware, software and networking expertise in a multivendor environment. https://www.pcmag.com/encyclopedia/term/52450/systems-integrator

269)

TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

Tools: Source Code Scanning and License Compliance

Return to Tools Area

Source: Tools for managing open source programs

- ***Antepedia Reporter*** *– A commercial, fee-based application from Antepedia, Reporter is a report-generation product which lets developers, project managers, legal advisors and others create license compliance audits and IP rights management reports about the open source, public and private components in your code base. http://www.antepedia.com/pages/tools.html*
- ***Black Duck Hub*** *– The commercial Hub service scans code to identify all embedded open source components, and then automatically searches for known vulnerabilities for remediation. It can send alerts when new vulnerabilities are found in your code. https://www.blackducksoftware.com/products/hub*
- ***Black Duck Protex*** *– Protex is a commercial, fee-based license compliance management tool from Black Duck which integrates with existing tools to automatically scan, identify and inventory open source software, while also enforcing license compliance and corporate policy requirements.https://www.blackducksoftware.com/products/protex*
- ***Copyright review tools*** *– This collection of open source command line tools help make initial copyright file construction and subsequent review and update easier. https://wiki.debian.org/CopyrightReviewTools*
- ***dep-checker*** *– A free dependency checker tool from The Linux Foundation, dep-checker performs a complete analysis of linkages between code packages. http://git.linuxfoundation.org/dep-checker.git/*
- ***FlexNet Code Insight*** *– Flexera, which acquired licensing compliance vendor Palamida in 2016, commercially offers Flex Code Insight to help automate corporate open source use among developers, legal teams and security staffers. https://www.flexerasoftware.com/enterprise/products/software-vulnerability-management/flexnet-code-insight/*
- ***FOSSA*** *– This is a commercial tool that automatically performs code dependency tracking, license compliance scanning in the background. http://fossa.io/*
- ***FOSSology*** *– A Linux Foundation project, FOSSology is an open source license compliance software toolkit which can run license, copyright and export control scans from the command line. A database and web UI are also available to create compliance workflows. https://www.fossology.org/*
- ***janitor.git*** *– Code Janitor is an open source tool that helps evaluate source code for compliance with open source licenses. From The Linux Foundation, Code Janitor can be used with other products to check code. http://git.linuxfoundation.org/janitor.git/*
- ***LicenseFinder*** *– An open source tool which detects the licenses of the code being used in your projects, compares those licenses against a user-defined whitelist and then provides an actionable report. https://github.com/pivotal/LicenseFinder*

- ***Protecode Enterprise Analyzer*** *– This commercial application is used to analyze and identify all code in any directory to determine code ownership and ensure open source license compliance based on predetermined internal policies.*
  http://www.protecode.com/our-products/system-4/enterprise-analyzer/

- ***scancode-toolkit*** *– From nexB, the open source ScanCode suite of utilities scans code for licenses, copyright and dependencies to find, discover and inventory open source and third-party components used in your code. https://github.com/nexB/scancode-toolkit*

- ***SPDX*** *– The Software Package Data Exchange (SPDX) specification is a standard format used to describe the components, licenses and copyrights associated with software packages. The SPDX standard aids compliance with free and open source software licenses by standardizing the way license information is shared between developers and companies. The SPDX specification is developed by the SPDX workgroup, which is hosted by The Linux Foundation. The group offers open source tools to help users of SPDX documents. https://spdx.org/tools*

- ***WhiteSource*** *– Provides licensing, security, code quality and reporting analysis for managing open source components in real-time by automatically and continuously scanning dozens of open source repositories on a commercial basis. https://www.whitesourcesoftware.com/*

Tools: Bug and Issue Tracking

Return to Tools Area

Source: Tools for managing open source programs

- ***Bugzilla*** *– Open source, server-based software featuring an advanced query tool that can remember searches, integrated email capabilities and a comprehensive permissions system. Bugzilla is used by Mozilla as its bug tracking system.https://www.bugzilla.org/*
- ***GitHub Issues*** *– GitHub's own integrated feedback and bug tracker, GitHub Issues is available as part of GitHub's project hosting.https://help.github.com/articles/about-issues/*
- ***GitLab*** *– This bug tracking tool unifies issue tracking, code review, Git repository management, activity streams, wikis and more in a single UI to assist your open source projects. GitLab is available as a service or as a commercial software.https://about.gitlab.com/*
- ***JIRA*** *– From Atlassian, JIRA contains custom filters, developer tool integrations, customizable workflows and rich APIs to integrate JIRA with other applications. JIRA is available as a commercial software. https://www.atlassian.com/software/jira*

Tools: Archiving and Release Management

Return to Tools Area

Source: Tools for managing open source programs

- ***Artifactory*** *– Artifactory is a repository manager from JFrog which supports software packages created in any code language. It integrates with all major DevOps and continuous integration and continuous deployment tools. Artifactory is available as a commercial or as an open source tool. https://www.jfrog.com/artifactory/*
- ***Bintray*** *– A commercial archiving tool from JFrog that allows companies to publish their code release archives to maintain storage for older and larger files. https://bintray.com/*
- ***Docker Hub*** *– A cloud-based registry service which allows users to link to code repositories and build and test their images. It also stores manually-pushed images and links to Docker Cloud so users can deploy images to project hosts. Docker Hub is a centralized resource for container image discovery, distribution and change management, collaboration and workflow automation throughout the development pipeline. https://hub.docker.com/*
- ***github-release*** *– The open source, built in functionality part of GitHub which lets users package and edit releases of projects on GitHub so they are available for use by other community members. https://github.com/aktau/github-release*

Tools: Tracking Project Health

Return to Tools Area

Source: Tools for managing open source programs

- ***CatWatch*** *– CatWatch is an open source metrics dashboard from Zalando that fetches GitHub statistics for your GitHub accounts, processes and saves your GitHub data in a database. The data reveals the popularity of your open source projects, your most active contributors and other interesting statistics.https://github.com/zalando-incubator/catwatch*
- ***Gander*** *– Gander is an open source dashboard which generates usable metrics for a range of open source projects in one quick look. Created by PayPal, Gander is designed for individuals who are responsible for running Open Source Program Offices or keeping track of multiple open source projects. https://github.com/paypal/Gander*
- ***GHCrawler*** *– Created by Microsoft, GHCrawler is an open source GitHub API crawler that crawls a GitHub-hosted project and automatically tracks, retrieves, and stores its contents. GHCrawler is primarily intended for people trying to track sets of organizations and data repositories. https://github.com/Microsoft/ghcrawler*
- ***Gittagstats*** *– Gittagstats is an open source tool which generates statistics reports from a set of tags for a Git repository. The tool was created by Qualcomm. https://github.com/mcharleb/gittagstats*
- ***GrimoireLab*** *– GrimoireLab has a variety of open source tools to measure open source project statistics and visualize them, from git repositories, GitHub pull requests or Bugzilla tickets to mailing lists, Meetup groups or Slack channels. GrimoireLab is a project in CHAOSS, a collaborative group on open source development metrics. https://grimoirelab.github.io/*
- ***OSS-dashboard*** *– The Open Source Program Dashboard, which comes from Amazon, is a multi-function dashboard which can be used to view and monitor many GitHub organizations and or users at one time. https://github.com/amzn/oss-dashboard*
- ***OSS Tracker*** *– OSS Tracker, from Netflix, collects data about a GitHub organization and aggregates it across all projects within that organization in a single user interface. All repositories are listed and metrics are combined for an organization, but community managers can also organize projects into functional areas and appoint administrators to assign management and engineering leads. https://github.com/Netflix/osstracker*

Tools: Code Reviews

Return to Tools Area

Source: Tools for managing open source programs

- ***mention-bot*** *– Developed by Facebook, this tool automatically mentions potential reviewers for code contributions by community members to speed up the review process.https://github.com/facebook/mention-bot*
- ***PullApprove*** *– Brings more formalization to code contributions – or pull requests – by improving code quality through peer-review, enforcing style guidelines, catching errors and providing security checks on code.https://about.pullapprove.com/*
- ***sentinel*** *– A repository management bot which reviews and tests code contributions, builds a list of maintainers for the repository and communicates the status of a pull request with users.https://github.com/habitat-sh/sentinel*

Tools: Contributor License Agreements (CLA)

Return to Tools Area

Contributor License Agreements (CLA) *is legal documentation that serves to protect any contributors to a project in regard to copyright issues and distribution. Essentially the CLA states that the contributor writer/computer programmer/etc. has agreed to allow their input to be used in the ongoing development of the project, while protecting them against future copyright infringement, and permitting them to take legal action should there be a case of infringement or should they not be receiving credit for their contributions. Additionally, it protects the other parties, as well, as the CLA does not allow for the contributor to later refuse use of their material.*
https://www.upcounsel.com/contributor-license-agreement

There are different tools to help with CLA[270)]

- ***CLA Assistant*** *– Contributed by SAP, the CLA Assistant streamlines workflows by handling the legal side of contributions for users. The Assistant asks code contributors to sign CLAs as they make their code contributions and authenticates each contributor with his or her GitHub account. It also updates the status of a pull request when the contributor agrees to the CLA and automatically asks users to re-sign the CLA for each new pull request if changes are made to the CLA.https://github.com/cla-assistant/cla-assistant*
- ***CLA Portal*** *– From VMware, CLA Portal adds a workflow to enable contributors to digitally sign a CLA for pull requests to your GitHub repositories. When a developer opens a pull request, they are prompted to sign the agreement if needed. Also included is an administrator interface for CLA authoring, CLA-to-project mapping, and agreement reviews.https://github.com/vmware/claportal*
- ***DCOB*** *– A Developer Certificate of Origin Bot which helps to enforce developer certificate of origin sign-offs for each code change in a pull request. The DCOB sets the status for each accepted code change, as required by the Developer Certificate of Origin.https://github.com/chef/dcob*

270)
Tools for managing open source programs

Tools: GitHub Management at Corporate Scale

Return to Tools Area

Source: Tools for managing open source programs

- ***hubcommander*** *- A Slack bot for GitHub organization management, HubCommander uses chat-ops – or conversation-driven development – to help manage GitHub projects. It creates a simple way to perform privileged GitHub organization management tasks without granting administrative or owner privileges to your GitHub organization members.https://github.com/Netflix/hubcommander*
- ***opensource-portal*** *– From Microsoft, this tool is designed to help large organizations with their large-scale GitHub management operations, onboarding and more. This is one of a suite of tools provided by the Open Source Programs Office at Microsoft.https://github.com/Microsoft/opensource-portal*
- ***settings*** *– This app syncs repository settings defined in .github/settings.yml to GitHub, enabling pull requests for repositories.https://github.com/bkeepers/github-configurer*
- ***zappr*** *- Zappr is a GitHub integration built to enhance project workflows. From Zalando, zappr helps developers to increase productivity and improve open-source project quality by removing bottlenecks around pull request approval and helping project owners halt “rogue” pull requests before they’re merged into the project master branches.https://github.com/zalando/zappr*

Tools: Project Quality

Return to Tools Area

Source: Tools for managing open source programs

- ***CII Best Practices Badging*** *– From The Linux Foundation, the Core Infrastructure Initiative (CII) Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify for free by using this web application to explain how they follow each best practice.https://bestpractices.coreinfrastructure.org/*
- ***CodeClimate*** *– Code Climate empowers organizations to take control of their code quality by incorporating fully configurable test coverage and maintainability data throughout the development workflow. It's free for open source projects!https://codeclimate.com/*

Tools: Logging Tools

Return to Tools Area

Source: Apache Logging Services

- ***Apache Chainsaw*** *is GUI based log viewer and is a companion application to log4j written by members of the log4j development community. Chainsaw can read local and ssh-reachable regular text log files, as well as log files formatted in Log4j's XMLLayout. Chainsaw can also receive events over TCP and UDP, read events from a DB, and can process events generated by java.util.logging. https://logging.apache.org/chainsaw.*
- ***Apache Log4j Audit*** *provides the information to determine what actions have been performed, or attempted to be performed, by whom, when they did it and what the data associated with the action was. Audit log data is most typically used by security teams to monitor for fraud or illegal or otherwise unauthorized activities and to be able to correct changes that were incorrectly made. Audit logs are often used to demonstrate compliance with legal obligations such as Sarbanes-Oxley Act or HIPPA.https://logging.apache.org/log4j-audit/latest/*

Tools: Network Traffic Analysis

Return to Tools Area

Network Traffic Analyzers are generally deployed within Enterprises. Although having many of the characteristics of a single, united Enterprise, DIDOs are naturally more of a federation or coalition of the willing spread across the internet. This is why virtualized testing is so important.

Source: The 14 Best Network Traffic Analysis Solutions for 2019 and Beyond, September 17, 2019, https://solutionsreview.com/network-monitoring/the-14-best-network-traffic-analysis-solutions-for-2019-and-beyond/

- ***Awake Security Platform*** *is a network traffic analysis solution that focuses on discovering, assessing, and processing security threats. The tool is broken down into three parts: Awake Sensors, which continuously monitor and collect data from devices, apps, and users; Awake Nucleus, which analyzes that data to understand behaviors and attributes of entities and applying deep forensics; and Ava, a privacy-aware security expert system that applies machine learning to collected data.*
- ***Corelight*** *is a security-focused network traffic analysis provider that uses the open source network security monitor Zeek as its basis. Corelight Sensors convert network traffic data into logs and extracted files which can all be managed through the Corelight Fleet Manager. Through the Fleet Manager, admins can define custom groups, assign individual roles, and set access levels. Corelight Sensors come either as hardware for networks, as a virtual sensor, or as a cloud traffic monitor for AWS.*
- ***Flowmon*** *is a network performance and security solution provider that offers network traffic monitoring and analysis capabilities. The solution offers real-time NetFlow and IPFIX monitoring and analyzes network traffic data from a physical, virtual, or cloud infrastructure. It also gathers flow data statistics generated by routers, switches, or standalone hardware probes. Users can add self-defined filters that set parameters for data collection based on what data the user wants to look at.*
- ***Kentik Platform*** *is an AIOps platform that applies artificial intelligence and machine learning capabilities to network traffic analysis. The solution analyzes downstream and transit traffic flows and helps enterprises identify peering opportunities, optimize their network routing, and gain more control over their service performance. They also offer network traffic engineering capabilities to maximize resource utilization and traffic delivery, and insights into network capacity to help drive cost-efficient traffic flow.*
- ***LogRhythm NetworkXDR*** *is a security-focused network traffic analysis solution that focuses on threat detection and analytics. It offers real-time network traffic analysis via network sensors that allow for distributed traffic data collection and reporting. The solution is designed to increase network traffic visibility with application identification, app-aware metadata, and full packet capture. NetworkXDR also integrates with LogRhythm's NextGen SIEM Platform to help identify security threats.*
- ***ManageEngine Netflow Analyzer*** *is a bandwidth monitoring tool that is built on network traffic monitoring and analysis functions. The program implements network flow analysis to examine bandwidth usage, network data, and traffic patterns. It condenses information about which users*

and devices are using available bandwidth on your network – as well as what they're using it for. The solution also feature network forensics and security features, application monitoring, and data capacity planning and billing capabilities.

- ***Netfort LANGuardian*** *is a network traffic analysis and packet inspection software that monitors network and user activity. LANGuardian uses packet inspection tools to troubleshooting bandwidth problems, create audit trails of file and folder activity, and examine Internet gateways. The solution uses wire data analytics to capture metadata from network packets, provides continuous health checks on network and user activity, and alerts admins to any suspicious data.*

- ***NETSCOUT*** *is a service assurance and network monitoring vendor that provides network traffic data inspection and analysis. The solution continuously inspects traffic data and analyzes large volumes of data through Layer 7/8 deep packet inspection, load balancing and acceleration, aggregation and desegregation, and packet decoding. NETSCOUT also utilizes their Adaptive Service Intelligence (ASI) technology that uses traffic data to gain visibility into user communities, services, and IT assets.*

- ***ntopng*** *is an open source network traffic probe and analysis tool. The traffic probe sorts network traffic into different criteria, including IP addresses and throughput. By characterizing network traffic, your enterprise can easily determine different network statistics that are affecting your network; the solution can reference real-time and historical traffic data in this analysis. While ntopng's Community version is open source, Professional and Enterprise versions are also available.*

- ***Paessler PRTG*** *is an IT monitoring tool that includes network traffic analysis functionality. PRTG's network traffic analysis system helps administrators track network capacity and seeing how much of their data analysis is actually being used. The solution combines SNMP monitoring, packet sniffing, and data flow technologies like NetFlow, IPFIX, jFlow, and sFlow for their traffic analysis capabilities; it displays traffic data alongside the other performance and security insights it uncovers.*

- ***Plixer Scrutinizer*** *is a network traffic analysis system that gathers network traffic flow and metadata across an entire network infrastructure. The solution collects data from SD-WAN, cloud, firewalls, routers, data centers, probes, data collectors, and wired/wireless edges. Scrutinizer then takes this data and provides valuable security and performance insights. This tool can help IT teams optimize network and application performance by providing end-to-end network visibility.*

- ***SolarWinds NetFlow Traffic Analyzer*** *is a NetFlow traffic analysis and bandwidth monitoring solution. The tool is designed specifically to analyze NetFlow traffic data as well as IPv4 and IPv6 flow records and application traffic. Users can also visually correlate performance and traffic data discrepancies by displaying metrics right next to each other. It also can integrate with SolarWinds' other Orion Platform products, such as their Network Performance Monitor and Network Configuration Manager.*

- ***Ipswitch WhatsUp Gold*** *is an all-in-one infrastructure monitoring tool that features network traffic analysis capabilities. WhatsUp Gold provides insight into application bandwidth usage and helps administrators to manage the performance of your infrastructure, applications, and services. It also leverages real-time and historical bandwidth usage data to help enterprises keep track of capacity, as well as determine what traffic was consuming bandwidth during a period of slow network performance.*

Appendix D: Acronyms

return to the Public Area **OR** return to Reference Architecture (RA)

Acronym	Meaning
AON	All-Or-None
API	Application Programming Interface
APP	Application
ASF	Apache Software Foundation
ASIC	Application-Specific Integrated Circuit
BFT	Byzantine Fault Tolerance
BIP	Bitcoin Improvement Proposal
CISQ	Consortium for Information & (or IT) Software Quality
CLA	Contributor License Agreements
CIL	Common Intermediate Language
CLI	Command Line Interface
CLR	Common Language Runtime
CM	Case Management
CM	Configuration Management
CoI	Community of Interest
CPU	Central Processing Unit
ĐApp	Đistributed Application
DApp	Distributed Application
DaaS	Data as a Service
dBFT	Delegated Byzantine Fault Tolerant
DBMS	Database Management Systems
DDS	Data Distribution Service
DIDO	Distributed Immutable Data Objects
DIL	Disconnected, Intermittent and Limited networks
DLT	Distributed Ledger Technology
DM	Data Model
DNS	Domain Name System
DoD	U.S. Department of Defense
DPoS	Delegated Proof of Stake
DSS	Digital Signature Standard
ECMA	European Computer Manufacturers Association
ECMAScript	European Computer Manufacturers Association Scripting Language Specification
ERC	Ethereum Request for Comment
EIP	Ethereum Improvement Proposal
EVM	Ethereum Virtual Machine
FIGI	Financial Instrument Global Identifier
FOK	Fill-Or-Kill

Acronym	Meaning
GDPR	General Data Protection Regulations
GMS	Google Mobile Services
GPU	Graphical Processing Unit
gRPC	Google Remote Procedure Call
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IA	Identification and Authentication
IaaS	Infrastructure as a Service
ICO	Initial Coin Offering
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIOT	Industrial Internet of Things
IP	Intellectual Property
IP	Internet Protocol
IPFS	Interplanetary File System
IS or InfoSec	Information Security
ISO	International Organization for Standards
IT	Information Technology
ITU	International Telecommunications Union
JiT	Just-in-Time
JVM	Java Virtual Machine
K8	K8s or Kuberenetes
KYC	Know-Your-Customer
LMT	Limit Order
LSB	Linux Standard Base
MA	Mission Assurance
MKT	Market Order
MIT	Market If Touched
MIT	Massachusetts Institute Of Technology
MQ	Message Queue(MQ)
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
ODM	Ontology Definition Metamodel
OMG	Object Management Group
OpenJS	open source JavaScript
OpenMAMA	Open Middleware Agnostic Messaging API
OS	Operating System
OT	Operational Transforms
OSI	Open Systems Interconnection
OSS	Open Source Software
OWL	Web Ontology Language

Acronym	Meaning
P2P	peer-to-peer
P&P	Policy and Procedure
PaaS	Platform as a Service
PIM	Platform Independent Model
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
Protobuf	Google Protocol Buffer
PSM	Platform Specific Model
RA	Reference Architecture
RDF	Resource Description Framework
RFP	Request For Proposal
RFC	Request for Comment
RFI	Request for Information
REST	Representational State Transfer
RDBMS	Relational Database Management Systems
RESTful	RESTful API
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SCAP	Security Content Automation Protocol
SfA	Safety Assurance
SPDX	Software Package Data Exchange
SPV	Simple Payment Verification
SQuaRE	Systems and software Quality Requirements and Evaluation
STP	Stop Order
STP	Straight-through Processing (StP)
STPLMT	Stop Limit Order
SW	Software
SwA	Software Assurance
SysA	System Assurance
SysML	System Modeling Language
TCP	Transmission Control Protocol
SI	System Integrator
SIG	Special Interest Group
SDO	Standards Developing Organization
SoS	System of Systems
SPARQL	Simple Protocol and RDF Query Language
SSO	Standards Setting Organization
TODO	Talk Openly Develop Openly
UAF	Unified Architecture Framework

Acronym	Meaning
UDP	User Datagram Protocol
UID	Unique Identifier
UUID	Universal Unique Identifier
UML	Unified Modeling Language
VM	Virtual Machine
W3C	World Wide Web Consortium
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XSLT	XSL Transformations
ZMTP	ZeroMQ Message Transport Protocol