**Planetary Data System**

NSSDCA Delivery Tool

Software Requirements and Design Document (SRD/SDD)



Jordan Padams

TBD

Version 1.0

****

Jet Propulsion Laboratory

Pasadena, California

**CHANGE LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Description** | **Author** |
| 0.1 | 2019-10-29 | Initial draft. | J. Padams |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**TABLE OF CONTENTS**

1.0 INTRODUCTION 4

1.1 Document Scope and Purpose 4

1.2 Method 4

1.3 Notation 4

1.4 Controlling Documents 5

1.5 Applicable Documents 5

1.6 Document Maintenance 5

2.0 Component Description 6

2.1 Archival Information Package (AIP) 6

2.2 Submission Information Package (SIP) 6

3.0 Use Cases 8

3.1 Generate Archive Information Package 8

3.2 Generate Submission Information Package 9

3.3 Track Status of Data Deliveries to Deep Archive 10

4.0 Requirements 11

4.1 Level 4 Requirements 11

4.2 Level 5 Requirements 11

5.0 DESIGN PHILOSOPHY, ASSUMPTIONS, AND CONSTRAINTS 13

6.0 ARCHITECTURAL DESIGN 14

6.1 Component Architecture 14

6.1.1 External Interface Design 14

6.1.2 Internal Interface Design 14

6.2 Data Model 14

7.0 ANALYSIS 15

8.0 IMPLEMENTATION 16

9.0 DETAILED DESIGN 17

APPENDIX A Acronyms 19

# INTRODUCTION

The PDS4 effort will overhaul the PDS data architecture (e.g., data model, data structures, data dictionary, etc) and deploy a software system (online data services, distributed data catalog, etc) that fully embraces the PDS federation as an integrated system while leveraging modern information technology.

Included in this overhaul is the software necessary to interface with the NASA Space Science Data Coordinated Archive (NSSDCA), as well as to meet the guidelines of the Open Archival Information System (OAIS) to ensure the integrity of the archive.

This software tool provides the functionality to generate the labels and artifacts necessary to ensure the PDS meets these requirements.

## Document Scope and Purpose

This document addresses the use cases, requirements and software design of the Registry service within the PDS4 data system. This document is intended for the reviewer of the service as well as the developer and tester of the service.

## Method

This combined Software Requirements and Software Design Document (SRD/SDD) represents the software by defining use cases and requirements and by using architecture diagrams, functional descriptions, context diagrams and data flow diagrams for the high-level design. UML diagrams will illustrate the detailed design.

## Notation

The numbering of the requirements in this document will be formatted as **LX.NSS.AA.X**, where:

* **LX** represents the requirements level where X is a number.
* **NSS** is an abbreviation representing the NSSDCA delivery software requirements section for the specified level.
* **AA** is a two-letter abbreviation representing the requirement sub-category (optional).
* **X** is a unique number within the section and optional sub-category for the requirement.

Following the text of a requirement may be a reference to the requirement or use case from which it was derived. The reference will be in parenthesis. A paragraph following a requirement, which is indented and has a reduced font size, represents a comment providing additional insight for the requirement that it follows. This comment is not part of the requirement for development or testing purposes.

## Controlling Documents

[1] Memorandum of Understanding: Planetary Data System and NASA Space Science Data Coordinated Archive, May 13, 2016.

[2] Planetary Data System (PDS) Level 1, 2 and 3 Requirements, March 26, 2010.

[3] PDS4 Project Plan, July 17, 2013.

[4] PDS4 System Architecture Specification, Version 1.3, September 1, 2013.

[5] PDS4 Operations Concept, September 1, 2013.

[6] Planetary Data System (PDS) General System Software Requirements Document (SRD), Version 1.1, September 1, 2013.

Note: All documents can be found at <https://pds-engineering.jpl.nasa.gov>

## Applicable Documents

[7] IASA Technical Committee, *Guidelines on the Production and Preservation of Digital Audio Objects*, ed. by Kevin Bradley. Second edition 2009. (= Standards, Recommended Practices and Strategies, IASA-TC 04). [www.iasa-web.org/tc04/audio-preservation](https://www.iasa-web.org/tc04/audio-preservation)

[8] PDS4 Information Model Specification, PDS4 Information Model Specification Team. <https://pds.nasa.gov/datastandards/documents/im/>

[9] PDS4-NSSDCA Interface Process, May 23, 2017. <https://pds-engineering.jpl.nasa.gov/sites/default/files/NSSDCA_SubmissionProcess_final_20170523.pdf>

[8] Harvest SRD/SDD (TBD)

[9] Registry SRD/SDD (TBD)

## Document Maintenance

The component design will evolve over time and this document should reflect that evolution. This document is limited to design content because the specification content will be captured in separate documentation (e.g., Installation Guide, Operation Guide, etc.). This document is under configuration control.

# Component Description

NSSDCA Delivery Tool is responsible for the generation of two information packages for ensuring the integrity of the archive with respect to the NSSDCA and Open Archival Information System (OAIS) reference model [6]. The packages are intended to be generated by Discipline Node staff any time new data is released into the PDS, but can also be executed retroactively for data that has not previously been submitted. Once generated, the Archive Information Package and Submission package are both validated by the PDS Engineering Node, and the Submission package is delivered to the NSSDCA.

The two packages are the Archive Information Package and Submission Information Package.

## Archival Information Package (AIP)

The following information is copied verbatim from the *Guidelines on the Products and Preservation of Digital Audio Object* [7]:

*6.3.1.1  The definition of the term Archival Storage in OAIS includes the services and functions necessary for the storage of theArchival Information Package (AIP). Archival storage encompasses data management and includes processes such as storage media selection, transfer of AIP to storage system, data security and validity, backup and data restoration, and reproduction of AIP to new media.*

*6.3.1.2  AIP, as defined in OAIS reference model (CCSDS 650.0-B-1 Reference Model for an Open Archival Information System (OAIS)), is an information package that is used to transmit archival objects into a digital archival system, store the objects within the system, and transmit objects from the system. An AIP contains both metadata that describes the structure and content of an archived essence and the actual essence itself. It consists of multiple data files that hold either a logically or physically packaged entity. The implementation of AIP can vary from archive to another archive; it specifies, however, a container that contains all the necessary information to allow long term preservation and access to archival holdings. The metadata model of OAIS is based on METS specifications.*

*6.3.1.3  From physical point of view the AIP contains three parts; metadata, essence and packaging information, which all consists of one or more files (see 6.1.3 Defining the Digital Object). Packaging information can be thought as wrapper information and it encapsulates metadata and essence components.*

## Submission Information Package (SIP)

The following information is copied verbatim from the *Guidelines on the Products and Preservation of Digital Audio Object* [7]:

*6.2.1.1 The SIP is an Information Package that is delivered to the repository and digital storage system for ingest. The SIP includes the audio data to be stored and all the necessary related metadata about the object and its content. Ingest, in the OAIS model, is the process that accepts the content and all its related metadata (SIP), verifies the file, extracts the relevant data and prepares the AIP for storage, and ensures that AIPs and their supporting Descriptive Information become established within the OAIS.*

*6.2.1.2 A digital repository and preservation system should be able to accept and validate an audio file. Validation is a process that ensures that the files which are being accepted into the digital storage system comply with the standards. Non standard files may become difficult to use in the future when current replay systems no longer exist. Tools exist for automated validation of file formats, and some open source solutions, like JHOVE (JSTOR/Harvard Object Validation Environment), are available and being further developed.*

TBD Diagram

Figure 1: TBD

# Use Cases

A use case represents a capability of the component and why the user (actor) interacts with the system. It should be at a high enough level so as not to reveal or imply the internal structure of the system. An actor is an object (e.g., person, application, etc.) outside the scope of the component but interacts with the component. This section captures the use cases for the NSSDCA Delivery Tool based on the description of the tool from the previous section as well as use cases defined in the NSSDCA Submission Process document (see references). These use cases will be used in the derivation of requirements for the tool. The following diagram details the use cases:

TBD Diagram

Figure 2: NSSDCA Delivery Tool Use Cases

The above diagram identifies the following actors (represented as stick figures):

**Discipline Node Operator (DN)**

This actor represents a portion of the PDS Discipline Node staff responsible for executing this software for completing the NSSD Submission Process.

**NSSDCA Operator (NSSDCA)**

This actor represents the NSSDCA representative responsible for receiving the NSSDCA Submission Information Package.

**Engineering Node Operator (EN)**

This actor represents a portion of the PDS Engineering Node team responsible for validating the NSSDCA Submissions.

The following sections detail the use cases identified in the above diagram.

## Generate Archive Information Package

The NSSDCA Delivery Tool is responsible for generating an Archive Information Package (Product\_AIP) for internal delivery to the PDS Engineering Node. This use case pertains to the Discipline Node Operator and Engineering Node Operator actors.

1. DN creates or is delivered a new Bundle that has been placed online for public access, also known as a data release.
2. DN executes Harvest software to ingest the metadata for the bundle into the Registry.
3. DN executes the NSSDCA Delivery Tool to generate AIP from the registry.
4. NSSDCA Delivery Tool accepts the user input, queries the registry for the metadata needed to generate an AIP, and outputs the necessary products.
5. DN submits AIP to EN for validation and ingestion into the central Registry.

Alternative: Offline AIP Generation

At Step 2, the DN does not execute Harvest to ingest metadata into the Registry.

1. DN creates or is delivered a new Bundle that has been placed online for public access, also known as a data release.
2. DN executes NSSDCA Delivery Tool to generate AIP from bundle/collection tables and, optionally, an input checksum manifest (to avoid duplication of checksums).
3. NSSDCA Delivery Tool accepts the user input for offline AIP generation, utilizes the metadata in the archival files to generate an AIP, and outputs the necessary products.
4. DN submits AIP to EN for validation and ingestion into the central Registry.

## Generate Submission Information Package

The NSSDCA Delivery Tool is responsible for generating an Submission Information Package (Product\_SIP\_Deep\_Archive) for delivery to the NSSDCA for deep archiving of PDS data. This use case pertains to the Discipline Node Operator, Engineering Node Operator, and NSSDCA Operator actors.

1. DN creates or is delivered a new Bundle that has been placed online for public access, also known as a data release.
2. DN executes Harvest software to ingest the metadata for the bundle into the Registry.
3. DN executes the NSSDCA Delivery Tool to generate SIP from the registry.
4. NSSDCA Delivery Tool accepts the user input, queries the registry for the metadata needed to generate an SIP, and outputs the necessary products.
5. DN submits SIP to EN for validation and ingestion into the central Registry.
6. EN submits SIP to NSSDCA for validation and ingestion into their system for deep archive.

Alternative: Offline AIP Generation

At Step 2, the DN does not execute Harvest to ingest metadata into the Registry.

1. DN creates or is delivered a new Bundle that has been placed online for public access, also known as a data release.
2. DN executes NSSDCA Delivery Tool to generate SIP from bundle/collection tables and, optionally, an input checksum manifest (to avoid duplication of checksums).
3. NSSDCA Delivery Tool accepts the user input for offline SIP generation, utilizes the metadata in the archival files to generate an SIP, and outputs the necessary products.
4. DN submits SIP to EN for validation and ingestion into the central Registry.
5. EN submits SIP to NSSDCA for validation and ingestion into their system for deep archive.

## Track Status of Data Deliveries to Deep Archive

The NSSDCA Delivery Tool will generate the necessary products that will be delivered to the PDS Engineering Node for validation and registration. This will enable EN to track the status of all deliveries of archival to the deep archive across the PDS system.

1. EN receives a SIP from a DN with new data ready for release.
2. EN will perform validation on the SIP to ensure it matches the expected data release.
3. EN will ingest the SIP metadata into the Registry in order to track the status of submissions to the deep archive.
4. EN will deliver the SIP to the NSSDC.

Alternative: SIP fails validation

At Step 2, if the SIP fails validation at EN, the product will be returned to the DN for correction.

# Requirements

The architecture definition phase of the PDS4 project resulted in the decomposition of the system into several elements [3]. The Registry service derives from the Catalog/Data Management element, which was derived from requirements 2.2.2 and 2.6 of the PDS Level 1, 2, and 3 Requirements document [1]. The following level 3 requirements are relevant to this service:

**2.2.2** PDS will track the status of data deliveries from data providers through the PDS to the deep archive

**2.8.3** PDS will provide standard protocols for locating, moving, and utilizing data, metadata and computing resources across the distributed archive, among PDS nodes, to and from missions, and to and from the deep archive

**4.1.5** PDS will meet U.S. federal regulations for preservation and management of the data through its Memorandum of Understanding (MOU) with the NASA Space Science Data Coordinated Archive (NSSDCA).

In addition to the level 4 and 5 requirements specified below, the NSSDCA Delivery Tool must also comply with the general system requirements related to procedures and API definitions highlighted in the General System Design [5].

## Level 4 Requirements

The level four requirements in PDS represent subsystem or component requirements at a high level. The following requirements pertain to the NSSDCA Delivery Tool:

**L4.REG.3** - The system shall register products of a data delivery into an instance of the registry. (2.2.2, 2.8.3)

**L4.NSS.1** - The system shall generate products required for submission of data to the NASA Space Science Data Coordinated Archive (NSSDCA). (4.1.5)

## Level 5 Requirements

The level five requirements in PDS represent subsystem or component requirements at a detailed level. The following requirements pertain to the Registry service:

**L5.NSS.1** - The tool shall provide a command-line interface for execution. (TBD mapping)

**L5.NSS.2** – The tool shall be capable of generating a valid Archive Information Package transfer manifest (Product\_AIP) and PDS4 XML label in accordance with the PDS4 Information Model. (TBD mapping)

**L5.NSS.3** – The tool shall be capable of generating a valid Submission Information Package manifest (Product\_SIP\_Deep\_Archive) and PDS4 XML label in accordance with the PDS4 Information Model. (TBD mapping)

**L5.NSS.4** – The tool shall be capable of generating product manifests from an online Registry Service. (TBD mapping)

**L5.NSS.5** – The tool shall be capable of generating product manifests by crawling a file system using the information contained within the specified bundle and collection products. (TBD mapping)

**L5.NSS.6** – The tool shall generate a product manifest based upon the specification of a bundle as input, either through the path to a bundle label or a bundle LIDVID and an online Registry Service to query for information.

**L5.NSS.7** – The tool shall include products in the manifests based upon the following criteria:

* Bundle (B) specified as input to the tool
* Primary Collections (C1, C2, C3) associated with that Bundle (B)
* Primary products associated each of those Collections (C1, C2, C3)

**L5.NSS.8** – The tool shall be capable of accepting a checksum manifest as input to use in place of Registry or on-demand product checksum generation.

**L5.NSS.9** – The tool shall only accept checksum manifests with the following checksum types: MD5, SHA-1, or SHA-256.

**L5.NSS.10** – The tool shall generate a PDS4 label for the Submission Information Package using the latest PDS4 Information Model (1.13.0.0+)

**L5.NSS.11** – The tool shall require a Submission Information Package manifest only contain products with valid logical identifiers according to the PDS4 Standards Reference.

**L5.NSS.12** – The Submission Information Package manifest shall be a tab-delimited table with one record per product and four fields per record containing the following fields: checksum value, checksum type, resolvable URL to data product, unique product lidvid for the associated product.

# DESIGN PHILOSOPHY, ASSUMPTIONS, AND CONSTRAINTS

The design of the software will revolve around the generate of the SIP and AIP products to be delivered to the NSSDCA and Engineering Node.

# ARCHITECTURAL DESIGN

TBD

## Component Architecture

TBD

### External Interface Design

TBD

### Internal Interface Design

TBD

## Data Model

TBD

# ANALYSIS

The software needed to generate an AIP or SIP based on the PDS4 Information Model does not exist prior to this implementation.

The tool intends to leverage the PDS4 Registry as much as possible in order to minimize duplication of effort by software components. The Harvest Tool crawls and ingests all metadata from the bundle, including generating checksums, so the preferable execution of the software is to use an online registry component. The ‘offline’ mode is intended only in extenuating circumstances.

# IMPLEMENTATION

The NSSDCA Delivery Tool is planned for delivery during Build 10b of the PDS4 Project Implementation.

The software will be developed using Python 2.7.x with implementation focused on the ability for easy migration to Python 3.x when Python 2.7.x is deprecated.

Coding Standards

The software will follow the PEP 8 Style Guide and standard.

Security

The software will adhere to the necessary security guidelines as outlined by MGSS Security Plan and guidelines.

# DETAILED DESIGN

## Manifest Generation Process

The basic process of generating the manifests follows:

(1) inspect the Bundle.xml to get the manifest of collection.xml files

-- the collection manifest is located in the Bundle.xml:

<Bundle\_Member\_Entry>

<lid\_reference>urn:nasa:pds:excalibur:data\_collection</lid\_reference>

<member\_status>Primary</member\_status>

<reference\_type>bundle\_has\_data\_collection</reference\_type>

</Bundle\_Member\_Entry>

<Bundle\_Member\_Entry>

<lid\_reference>urn:nasa:pds:excalibur:context\_collection</lid\_reference>

<member\_status>Primary</member\_status>

<reference\_type>bundle\_has\_context\_collection</reference\_type>

</Bundle\_Member\_Entry>

<Bundle\_Member\_Entry>

<lid\_reference>urn:nasa:pds:excalibur:xml\_schema\_collection</lid\_reference>

<member\_status>Primary</member\_status>

<reference\_type>bundle\_has\_schema\_collection</reference\_type>

</Bundle\_Member\_Entry>

(2) inspect each Collection.xml to get the manifest for each product

-- the product manifest is located in the Collection.tab:

- 2 column CSV file

- member\_status

- LIDVID

S,urn:nasa:pds:context:target:exoplanet\_system.55-cnc::1.0

S,urn:nasa:pds:context:target:exoplanet\_system.gj-1132::1.0

S,urn:nasa:pds:context:target:exoplanet\_system.gj-1214::1.0

(3) compile all products in the Bundle (e.g., Bundle, Collections, basic products) into the manifests

- using the above exceptions

(4) XML labels

-- xml labels are to be generated against the current (i.e., most recent) version of the IM

(5) SIP\_4-column\_manifest

- see enclosure for sample SIP\_4-column\_manfiest XML label

- file reference to '4 column <tab> delimited table'

- 4 column <tab> delimited table:

- checksum\_value

- checksum\_type

- online accessible location (URL)

- LIDVID

- using the above sample\_bundle; manifest would contain:

66c540c106af2e94f639b0aa1c86d73d MD5 http://.../mission\_bundle/LADEE\_Bundle\_1101.xml urn:nasa:pds:ladee\_mission\_bundle::1.0

86f354fc7e5b02d9a1257d2f02779b44 MD5 http://.../mission\_bundle/context/collection\_mission\_context.xml urn:nasa:pds:ladee\_mission:context\_collection::1.0

d25b26e82ce39c3fe7be679536e246b6 MD5 http://.../mission\_bundle/context/collection\_mission\_context\_inventory.tab urn:nasa:pds:ladee\_mission:context\_collection::1.0

c49f24dfc6c0d9d8b55f73efd8e9657b MD5 http://.../mission\_bundle/document/collection\_mission\_document.xml urn:nasa:pds:ladee\_mission:document\_collection::1.0

46c500c7b3641747a13b1e33c525085a MD5 http://.../mission\_bundle/document/collection\_mission\_document\_inventory.TAB urn:nasa:pds:ladee\_mission:document\_collection::1.0

0346c3fba8e5f3cbe031d2de77c9372f MD5 http://.../mission\_bundle/document/ladee\_mission\_rev1\_5.pdf urn:nasa:pds:ladee\_mission:document:ladee\_mission::1.5

5ccf23968eee5e3518db488ea5db54e0 MD5 http://.../mission\_bundle/document/ladee\_mission\_rev1\_5.xml urn:nasa:pds:ladee\_mission:document:ladee\_mission::1.5

cbacf719a2fd4a6d1cf7d31ca900e199 MD5 http://.../mission\_bundle/document/ladee\_spacecraft\_rev1\_2.pdf urn:nasa:pds:ladee\_mission:document:ladee\_spacecraft::1.2

64163c0e513e268163fa8f5c931facdf MD5 http://.../mission\_bundle/document/ladee\_spacecraft\_rev1\_2.xml urn:nasa:pds:ladee\_mission:document:ladee\_spacecraft::1.2

2466f69593b27f0ef575b534ec8f6b78 MD5 http://.../mission\_bundle/xml\_schema/collection\_mission\_xml\_schema.xml urn:nasa:pds:ladee\_mission:xml\_schema\_collection::1.0

5dc1469ab53dfd726b0233cfa03c697f MD5 http://.../mission\_bundle/xml\_schema/collection\_mission\_xml\_schema\_inventory.tab urn:nasa:pds:ladee\_mission:xml\_schema\_collection::1.0

b3a6c88ebb369993bd6e3737b1accda3 MD5 http://.../mission\_bundle/xml\_schema/ladee\_1100.xml urn:nasa:pds:ladee\_mission:xml\_schema:ladee\_1100::1.1

98946dc2ccd1613a86a8cdef8aa48203 MD5 http://.../mission\_bundle/xml\_schema/ladee\_1100.xsd urn:nasa:pds:ladee\_mission:xml\_schema:ladee\_1100::1.1

(6) AIP\_manifest

- see enclosure for sample AIP\_manfiest XML label

- file reference to 'AIP checksum manifest'; uses <parsing\_standard\_id>MD5Deep</parsing\_standard\_id>

66c540c106af2e94f639b0aa1c86d73d LADEE\_Bundle\_1101.xml

86f354fc7e5b02d9a1257d2f02779b44 context/collection\_mission\_context.xml

d25b26e82ce39c3fe7be679536e246b6 context/collection\_mission\_context\_inventory.tab

c49f24dfc6c0d9d8b55f73efd8e9657b document/collection\_mission\_document.xml

46c500c7b3641747a13b1e33c525085a document/collection\_mission\_document\_inventory.TAB

0346c3fba8e5f3cbe031d2de77c9372f document/ladee\_mission\_rev1\_5.pdf

5ccf23968eee5e3518db488ea5db54e0 document/ladee\_mission\_rev1\_5.xml

cbacf719a2fd4a6d1cf7d31ca900e199 document/ladee\_spacecraft\_rev1\_2.pdf

64163c0e513e268163fa8f5c931facdf document/ladee\_spacecraft\_rev1\_2.xml

2466f69593b27f0ef575b534ec8f6b78 xml\_schema/collection\_mission\_xml\_schema.xml

5dc1469ab53dfd726b0233cfa03c697f xml\_schema/collection\_mission\_xml\_schema\_inventory.tab

b3a6c88ebb369993bd6e3737b1accda3 xml\_schema/ladee\_1100.xml

98946dc2ccd1613a86a8cdef8aa48203 xml\_schema/ladee\_1100.xsd

- file reference to 'AIP transfer manifest'

- 2 colummn fixed-width table:

- LIDVID (255 characters)

- file\_specification\_name (255 characters)

urn:nasa:pds:ladee\_mission\_bundle::1.0 \LADEE\_Bundle\_1101.xml

urn:nasa:pds:ladee\_mission:context\_collection::1.0 \context\collection\_mission\_context.xml

urn:nasa:pds:ladee\_mission:context\_collection::1.0 \context\collection\_mission\_context\_inventory.tab

urn:nasa:pds:ladee\_mission:document\_collection::1.0 \document\collection\_mission\_document.xml

urn:nasa:pds:ladee\_mission:document\_collection::1.0 \document\collection\_mission\_document\_inventory.TAB

urn:nasa:pds:ladee\_mission:document:ladee\_mission::1.5 \document\ladee\_mission\_rev1\_5.pdf

urn:nasa:pds:ladee\_mission:document:ladee\_mission::1.5 \document\ladee\_mission\_rev1\_5.xml

urn:nasa:pds:ladee\_mission:document:ladee\_spacecraft::1.2 \document\ladee\_spacecraft\_rev1\_2.pdf

urn:nasa:pds:ladee\_mission:document:ladee\_spacecraft::1.2 \document\ladee\_spacecraft\_rev1\_2.xml

urn:nasa:pds:ladee\_mission:xml\_schema\_collection::1.0 \xml\_schema\collection\_mission\_xml\_schema.xml

urn:nasa:pds:ladee\_mission:xml\_schema\_collection::1.0 \xml\_schema\collection\_mission\_xml\_schema\_inventory.tab

urn:nasa:pds:ladee\_mission:xml\_schema:ladee\_1100::1.1 \xml\_schema\ladee\_1100.xml

urn:nasa:pds:ladee\_mission:xml\_schema:ladee\_1100::1.1 \xml\_schema\ladee\_1100.xsd

###### Acronyms

The following acronyms pertain to this document:

TBD