

Configuration Persistence Service Project

- PROJECT NAME
- PROJECT DESCRIPTION
 - SCOPE
 - SEED CODE
- ARCHITECTURE ALIGNMENT
 - ARCHITECTURE CPS FLOW
- DEPLOYMENT VIEW
- CPS SECURITY REQUIREMENTS
- SECURITY ASSURANCE
- OPERATIONAL DESCRIPTION
 - BASIC CONCEPTS & PRINCIPLES
 - SETUP - INITIAL CONFIGURATION & SYNCHRONIZATION OF DB
 - SETUP - DB SCHEMA
 - ACCESS - RECEIVE INFORMATION
 - ACCESS - WRITING & READING INFORMATION & DATA PERSISTENCY
 - ACCESS - PUBLISHING UPDATES
 - OPERATIONAL - SYNCHRONIZATION
 - OPERATIONAL - SCALABILITY
 - OPERATIONAL - PERFORMANCE
 - OPERATIONAL - DATABASE MANAGEMENT FUNCTIONS
 - OPERATIONAL - DATA INTEGRITY & RECOVERY
 - ACCESS - CONSUMER API
- DEPENDENCIES
 - ONAP PROJECT DEPENDENCIES
 - OPEN SOURCE DEPENDENCIES
- RESOURCES
 - Table of Committers: (+2)
 - Table of Contributors: (+1)
- KEY PROJECT FACTS
- TSC APPROVAL
- ASSOCIATED FILES
- USE CASES
- ROADMAP
- NEXT STEPS (Incubation)

approved by TSC 08 Oct 2020 <https://lists.onap.org/g/onap-tsc-vote/topic/77246028#1868>

PROJECT NAME

- Proposed name for the project: ConfigPersistencySvc
- Proposed name for the repository: ConfigPersistencySvc
- Project Logo:



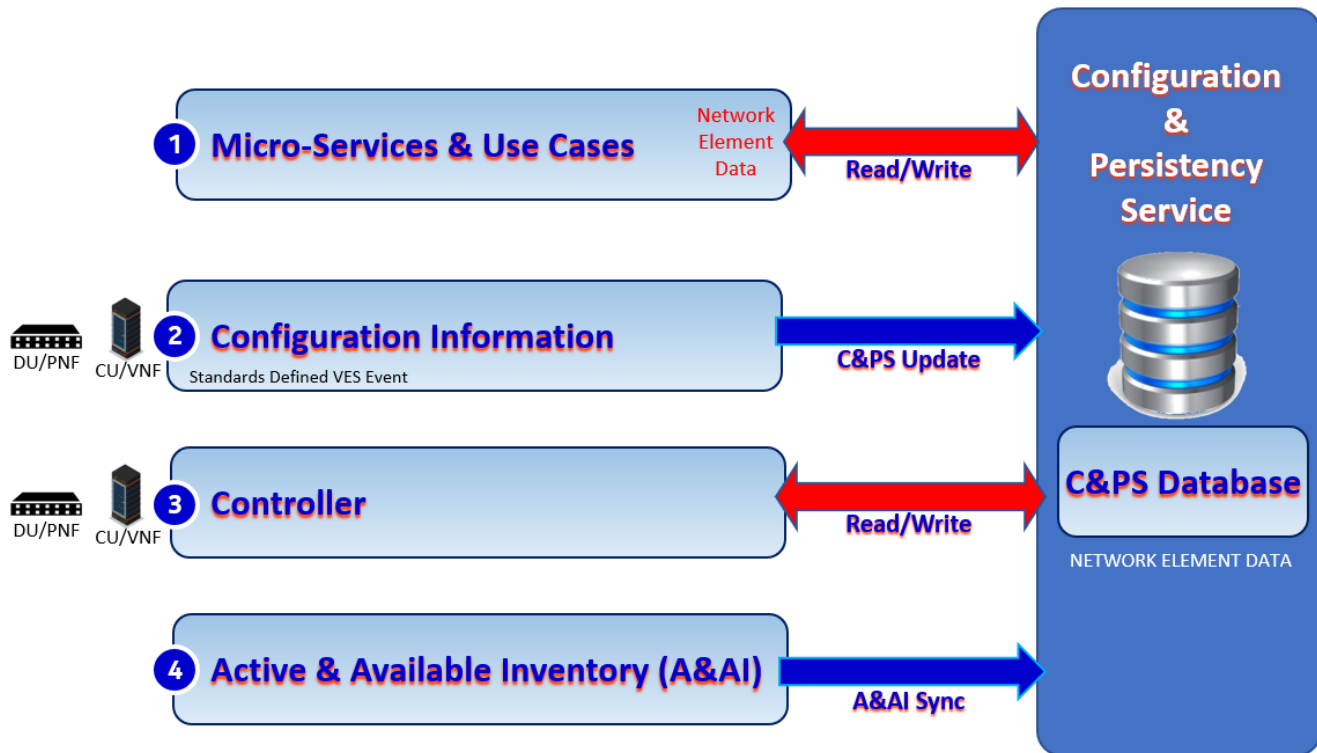
PROJECT DESCRIPTION

The Configuration Persistence Service is a platform component that is designed to serve as a data repository for Run time data that needs to be persistent. It is characterized by the following purpose statements:

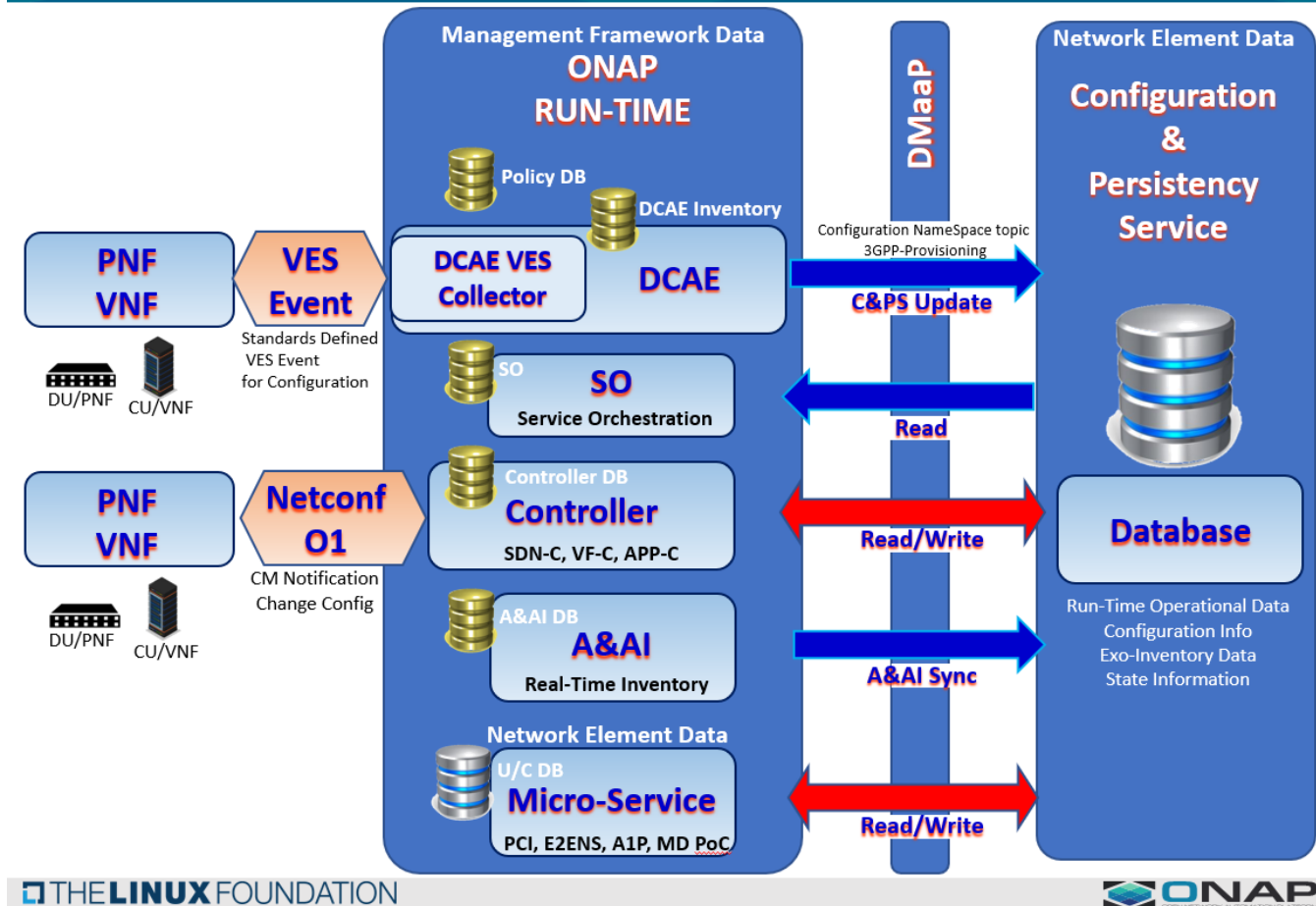
PURPOSE OF CONFIGURATION PERSISTENCE SERVICE:

- **REPOSITORY** - The types of data that is stored in the Run-Time data storage repository for:
 - (1) **CONFIGURATION PARAMETERS** - These are configuration parameters that are used by xNFs during installation & commissioning. Configuration parameters are typically used before the xNF has been brought up or is operational. For example, a 5G Network configuration parameter for a PNFs that sets the *mechanical* tilt which is a configuration setting upon installation.
 - (2) **OPERATIONAL PARAMETERS** - These are parameters that are derived, discovered, computed that are used by xNFs during run time AFTER the xNF becomes operational i.e. AFTER it has "booted up", been installed, configure. For example, in 5G Network, 5G PNFs may need to adjust a tower *electrical* antenna tilt. These operational parameters are Exo-inventory information, meaning it is information that doesn't belong in A&AI. In principle, some parameters might be both configuration and operational parameters depending on how they are used.
 - (3) **POLICY INFORMATION** - This type of information stores policy information that micro-services might need.
 - (4) **APPLICATION INFORMATION** - Information related to operational parameter.
- **DATA LAKE** - Architecturally, the Configuration Persistence Service is designed to be a common services data layer which can serve as a **data lake** to other run time entities (ONAP components or external tools).
- **C&PS FUNCTIONS** - The Configuration Persistence Service enables functionality to be performed by other entities. It will ENABLE the capability of another components or external tools within/or external to ONAP to perform the functions.
 - **Configuration Management FUNCTIONS** - Enables OSS configuration, optimization, and LCM operations.
 - **SYNCING** - The Configuration Persistence Service enables the ability to sync data between initial and delta changes ONAP & the xNFs.
 - **DATA RECOVERY** - It will allow for the recovery of data once a source of truth is defined.
 - **UPDATES** - It will allow for updates, delta changes, and incremental updates to be performed.
 - **RESTORATION** - It will allow for the recovery and restoration of data to a fall back point.
 - **DATA HISTORY** - It will allow an operator to see a history of changes in data. This will includes the versioning of the data, which is also associated with updates, roll back, restoration. Time series management is associated with data history management.
 - **AUDITING** - It will allow for auditing of configuration and parameters against a "golden" template. It itself stores & provides the data for another thing to perform the auditing. Consistency checks.
 - **ROLL BACK** - It will allow for rollback back to "save points" or recovery points.
 - **ACCESS CONTROL** - A security topic, which allows the definition of policies, of who and what they can update.
 - **TOPOLOGY TRAVERSAL** - It will enable the ability for something to traverse the relationship between data elements.
 - **MODEL ADAPTION** - Depending on the schemas used to define the DB data elements; it allows for the adaptation or transformation of models from other sources (e.g. ORAN or 3GPP models).
- **CENTRAL/DISTRIBUTED** - Because it is a common service, it is part of an ONAP installation, so it could be deployed with either an Edge ONAP installation or a centralized ONAP installation. The DB shall be designed to accommodate distributed operation as part of its role in the ONAP installation.

Configuration & Persistency Service (C&PS)



Configuration & Persistency Service (C&PS)



SCOPE

Functionality supported by a Configuration Persistence Service

The Configuration Persistence Service does several functions:

- RECEIVE INFORMATION** - This is the ability to receive information from a source, for example DMaaP, which contains information that needs to be incorporated into the data base.
- WRITE INFORMATION** - Configuration, operational and policy information is written into the database. Data received is written to the database via database write operations.
- PUBLISH CHANGES** - When changes occur to the database, it shall have the capability to publish changes. This will take the form of notifications on the DMaaP bus to allow other subscribers to become aware of changes.
- REFERENTIAL INTEGRITY** - When element updates occur, the component shall be able to maintain referential integrity. If elements are removed from the database, then associations or connections between elements may change and need to be updated.
- INGEST PACKAGES** - SDC CSAR parser - Get SDC CSAR Service Package - getting key artifacts from vendor provided onboarded packages (CSAR).
- LOGICAL OBJECTS** - The database shall support logical objects that are not PNFs, VNFs, or ANFs. These are objects that are elements that may logically represent entities that may need to be kept track of. An example might be a cell (carrier-sector) within a wireless RAN network.
- ASSOCIATIONS** - These are "linkages" between elements within elements (or data records) in the database. So this database is a *relational* database in the connective sense of a relational database (as opposed to the composition sense of a relational database).
- CARDINALITY RULES** - This allows for the specification of a cardinality of element associations.
- LINKING RESTRICTIONS** - There may be rules which allow you to specify restrictions on the kind of associations that can be made.
- SYNCHRONIZATION** - Real-time Synchronization allows for the database to reflect the state of the network. This also includes syncing with A&AI elements. A&AI is a real-time view of the resources and services available to ONAP to use. Thus, when a xNF gets created or deleted, these changes must be reflected also in the database.
- HEURISTICS ACCESS RULES** - There could be some rules or heuristics to allow particular data to be written and to control access, and who may have access.
- DATA INTEGRITY & RECOVERY** - within scope of the project are aspects of integrity & recovery such as data recovery, backup & export, restoration, database roll-back and data history

SEED CODE

Configuration Persistence Service code will be based on seed code implementing the design tool and cockpit.

The current functionality of this component includes:

Design

Runtime

- In R6 the RTCfgDB started in CC-SDK. Seed code was done in

CCSDK-1963 - Getting issue details...

STATUS

.

ARCHITECTURE ALIGNMENT

- SHARED SERVICE** - The diagram below shows how the Configuration Persistence Service project fits into the overall ONAP architecture as a shared (common) service serving as a data layer for other ONAP components.
- A&AI INDEPENDENCE** - The Configuration Persistence Service stores information that is conceptual distinct from A&AI. The data is related because the DB also needs to keep track of the existence of xNFs; and A&AI is the master of resources & services. For more information about how information from A&AI is "synced" with the Configuration Persistence Service see the Operational Description section below.
- DATA LAYER** - Architecture conclusions is that this component should serve as a data layer to other components and be a data lake serving as a repository for storing run-time information. It shall also be within the shared services part of the architecture. Architecture concluded that this project should be an independent component in ONAP platform.
- SCOPE (Type of Info Stored)** - The type of information stored in the Configuration Persistence Service is run-time configuration, operational, and policy information.
- ARCHITECTURE CONSISTENCY** - The project is consistent with a data-centric ONAP architecture.

The following are links to Architecture related pages that describe the flows and component interfaces:

Architecture Info	Wiki Links
COMPONENT DESCRIPTION R6	ARC RunTime DB Component Description - R6 Frankfurt
COMPONENT DESCRIPTION R7	ARC Configuration & Persistency Service (CnPS) Component Description - Guilin (R7) Release
PROJECT PROPOSAL	RunTime Config DB Project Proposal (Oct 25 2019)

ARCHITECTURE CPS FLOW

The following page contains the CPS flows.

The flows detail the information flows in particular uses of CPS.

ARCHCOM: [InfoFlow - Configuration Persistence Service \(CPS\) Information Flow](#)

DEPLOYMENT VIEW

Error rendering macro 'excerpt-include'

No link could be created for 'CPS-78: Deployment View'.

CPS SECURITY REQUIREMENTS

HTTPS

- CPS is compatible and in the process of migrating to a service mesh via the ONAP service mesh implementation. (<https://gergit.onap.org/r/c/oom/+124287>)

Logging and Monitoring

- CPS uses spring boots Logback and Log4j for logging information without exposing credentials (usernames and passwords).
 - CPS has no logging of sensitive information such as usernames and passwords in plain text. The log files are only accessible within the authorized users of the application deployment.

SECURITY ASSURANCE

Input Validation

- CPS uses spring boot input validation support for initial input validation.
- CPS uses java mechanisms to further validate inputs such as parameters.
- CPS accepts models and data which are validated via a third-party tool (OpenDayLight YANG parser).

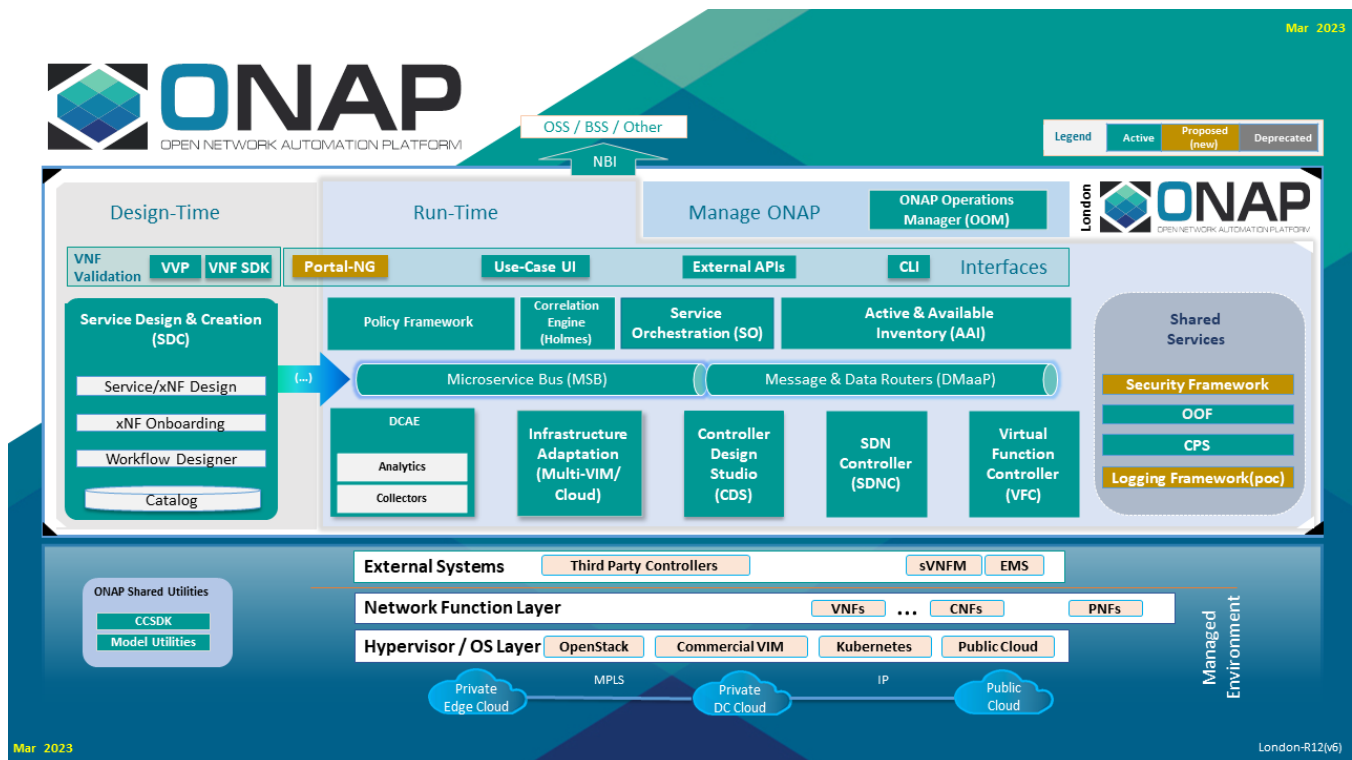
Authentication and Authorization

- CPS uses Basic Authentication control provided by spring security
- Usernames and passwords are configurable by the clients via passing the environment variables for use in application.yml file.
 - For deployments, CPS uses K8s secrets which are generated and stored as the application is deployed.
 - When CPS is run with docker, the services use usernames and passwords that are stored as environment variables.
 - CPS does not run docker containers or services as 'root'.

OPERATIONAL DESCRIPTION

C&PS DB operation is described. The following sections describe the basic operations of C&PS DB.

BASIC CONCEPTS & PRINCIPLES



- **ASSOCIATIONS** - These are "linkages" between elements within elements (or data records) in the database. Thus, this database is a *relational* database in the connective sense of a relational database (as opposed to the composition sense of a relational database).
- **CARDINALITY RULES** - This allows for the specification of a cardinality of element associations. For example, one PNF might have a limit to the number of associated logical elements, such as a Logical Cell that it is allowed to have.
- **LINKING RESTRICTIONS** - There may be rules which allow an operator to specify restrictions on the kind of associations that can be made. For example, an operator might want a particular kind of PNF to link to a specific kind of Logical object, such as a cell.

- **DATA LAYER** - This project is meant to serve as a data layer to other ONAP components. This means that it will be an intermediary for a component to write and access data.
- **PERSISTENCY** - The Configuration Persistence Service is meant to store data persistently, which means that it can hold data over time without losing it.
- **SYNCHRONIZATION** - The concept of synchronization is the ability to align data between the database and something else, such as an external source, or a xNF. For example, the Configuration Persistence Service needs to synchronize to A&AI view of the available resources in the system. *See the section on Synchronization below.*
- **STORAGE vs OWNERSHIP** - The Database *STORES* data, provides *persistence*, and gives access to data. The information is created, defined and used by other ONAP components. The *OWNERSHIP* and life cycle management is the responsibility of that other ONAP component. The DB stores the data but does not own the data. The result is that other ONAP components can freely access that data without other components creating new APIs. Thus, components don't have to have own data rather the database serves as steward of the data. If the owner is the only persona that writes the data, there should be no race conditions of two entities trying to write or modify data.
- **ACCESS CHARACTERISTICS** - *Historical data* and *current data* do not necessarily share the same characteristics & requirements. There might be multiple data base technologies that underlie the operation of the service.

SETUP - INITIAL CONFIGURATION & SYNCHRONIZATION OF DB

- **A&AI GETALL** - A&AI provides a "getall" function which will allow the DB to access a initial view of all of the xNFs in ONAP currently, rather than getting each of the xNFs individually from separate query requests.
- **INITIAL VALUES** - For each record that is setup, the individual attendant parameters associated with each record must also have some initial value. There are three cases:
 - (a) Some values may come from A&AI.
 - (b) Some values will be discovered during operation.
 - (c) Some information data may have default values defined in the schema.
- **INITIAL SYNCHRONIZATION** - The DB may also need to seek initial values for some of the parameters within the records, notably, those that need to come from a xNF, or external source (to ONAP). The mechanical tilt of an base station antenna value for example that value needs to come the source.
- **LOADING SCHEMAS** - These schemas would be loaded by an operator. There will be an initial way that it can get access to those schemas. Some schemas might be derived from a vendor package, or other information from an ONAP component. Overlapping information from A&AI will also be incorporated into the schema.
- **INGEST PACKAGES** - The DB must be able to get the output SDC CSAR Service Package. This will allow it to get key artifacts from vendor provided onboarded CSAR packages.
- **LOGICAL OBJECTS** - The database shall support logical objects that are not PNFs, VNFs, or ANFs. These are objects that are elements that may logically represent entities that may need to be kept track of. An example might be a cell (carrier-sector) within a wireless RAN network.
- **MODEL ADAPTION** - Depending on the schemas used to define the DB data elements; it allows for the adaptation or transformation of models from other sources (e.g. ORAN or 3GPP models).

SETUP - DB SCHEMA

- **MODEL DRIVEN** - The RunTime DB is driven by its schemas. In this case the schema is the model, and it adapt the capabilities of the RunTime DB. It clarifies the type of data and relationships of those data that is stored in the DB. The schema define type of data information, its structures and relationships.
- **PREDEFINED & DYNAMIC SCHEMA** - The pre-defined schemas are basic mirroring of the *topology* (meaning the existence of specific xNFs and their relationship to each other) from A&AI. By topology information, the DB must have a view of the xNFs and their relationship to each other, without duplicating the specific parameters that are stored in A&AI. An operator could define manually, in a predefined manner, the data and relationships that the DB managed. A dynamic schema would be automatically composed from other sources, such as artifacts from vendor onboard packages.
- **RULES** - Rules can also be specified in the schema, and then enforced by the model.

ACCESS - RECEIVE INFORMATION

- **RECEIVE INFORMATION** - This is the ability to receive information from a source, for example DMaaP, which contains information that needs to be incorporated into the data base. For example, a CMnotify VES event might come in from a PNF which holds a value the needs to be updated in the DB.
- **TRANSACTIONS** - The writing of data is an atomic transaction. A transaction is defined as an operation that is coherent and reliable. It is independent of other transactions. A transaction for the RunTime Config DB results in a consistent view of the database. All changes to a database happen *within* a transaction. A transaction applies to the "*Receive Information*", "*Writing information*" sections.

ACCESS - WRITING & READING INFORMATION & DATA PERSISTENCY

- **WRITE INFORMATION** - Configuration, operational and policy information needs to be written into the database. Data received is written to the database via database write operations.
- **DATA PERSISTENCY** - Once data is written to the database, it must persist over time. Data must be kept with integrity over the life of the operation of the instance of the RunTime Config DB.
- **WRITE NOTIFICATIONS** - Write failures should result in notifications. Obviously this should be a rare occurrence.
- **TRANSACTION** - *See above* for a definition of transactions as it relates to the RunTime Config DB.
- **TOPOLOGY TRAVERSAL** - It will enable the ability for users and other systems to traverse the relationship between data elements in the database.
- **RACE CONDITION AVOIDANCE** - Some mechanism must exist to prevent that two actors are updating the same piece of data simultaneously.
- **TIME SERIES DATA** - Service providers may wish to store and analyze time-series data. Time-series data are measurements collected from an entity over intervals of time. There will be metrics associated with the time-series data and an interface to collect those measurements over time. This has applications such as KPI analysis (PM LCM operations).

ACCESS - PUBLISHING UPDATES

- **PUBLISH UPDATES** - When changes occur to the database, it shall have the capability to publish changes. This will take the form of notifications on the DMaaP bus to allow other subscribers to become aware of changes.
- **SUCCESSFUL UPDATES** - When data is updated successfully, an "ACK" takes the form of publishing updates. Which allows a subscriber to become aware of a successful update. For example, if a Micro-service requested a PCI value to be changed from a 10 to 20; the DB would publish that this PCI value has changed once it successfully updated the database.

OPERATIONAL - SYNCHRONIZATION

- **REFERENTIAL INTEGRITY** - When element updates occur, the component shall be able to maintain referential integrity. If elements are removed from the database, then associations or connections between elements may change and need to be updated. A&AI is the ONAP master of inventory, if xNF/elements are removed or added the RunTime Config DB indices need to be updated.
- **A&AI SYNCHRONIZATION** - Real-time Synchronization allows for the RunTime Config database needs to reflect the current state of the network. This also includes syncing with A&AI elements. A&AI is a real-time view of the inventory, available resources and services available for ONAP to use. A&AI is the master of inventory information. Thus, when a xNF gets created or deleted, these changes must be reflected also in the RunTime Config database only insofar as it must also keep an "index" to existing xNFs to associated data with.
- **SYNCHRONIZATION OTHER SOURCES** - The database must also be able to Synchronization with external sources, such as PNFs. The RunTime Config DB shall have the ability to synchronize data between initial or delta changes between ONAP & an xNFs. For example, if a set of PCI values have changed, the database must be able to update itself to come into alignment with those changes.
- **PERIODIC SYNCHRONIZATION** - If there are notifications that are missed, the DB should periodically synchronize to make sure it is up to date. The operator should have some configurable period or strategy for synchronizing, for example, once a day. There should be some level of autonomy for automatically kicking off a periodic synchronization.

OPERATIONAL - SCALABILITY

- **SCALABILITY** - When there is a huge data set, for example in a 5G network with many PNF. Thus, it is important to consider scalability. The choice of database technology is important here. With containerized services, hosted in ONAP, there can't be a vertically growing service.
- **CLUSTERING** - Each DB instance work in a clustered manner. Many requests may arrive in parallel, and requests should be able to satisfied in Parallel.
- **PARTITIONING** - Partitioning is another possible strategy for horizontal scaling. This is one possible mechanism for enabling scalability.
- **REPLICATION** - Partitioning is another possible strategy for scaling. This is one possible mechanism for enabling scalability.
- **ELASTICITY** - To scale dynamically based on current experienced load. Load meaning amount of data, users and transactions.

OPERATIONAL - PERFORMANCE

- **PERFORMANCE** - The performance considerations for the DB might include aspects such as the number of incoming requests, number of concurrent requests that can be processed, prioritization of requests during congestion, and volume of returned data.
- **SCALABILITY** - Performance becomes particularly crucial to consider with respect to scalability, and it might be the case that different strategies are used based on the the scaling topics (see above).
- **NUMBER OF API REQUESTS** - Number of API requests that needs to be supported is a performance concern. The number of requests will depend on the ONAP configuration, whether there are edge deployments, and the capabilities of the cores that ONAP is deployed with. #@#
- **CONCURRENCY** - The number of concurrent requests that can be processed is also a Performance concern. The number of parallel requests will affect the amount of other requests that can be handled by other platform components and the messaging bus. A target number of concurrent API requests is #@#
- **RETURN SIZE** - The data size of queries to be returned is also a performance issue as it directly affects the amount of traffic that needs to be handled. We expect this to be model-driven which will be based on the schema used. For performance analysis purposes, a large worst-case estimates can be used. Though any estimate needs to take into account that there will be variability in the actual implementation. #@#
- **FORMAT** - How the query data is returned is important to consider. The data might be sent in binary format, JSON, or Zip format.
- **PRIORITIZATION** - During times of congestion, there might need to be a strategy for prioritizing requests, both incoming and return responses. This will be also tied into the scalability solution. One possibility might be to limit the number of requests per consumer API if that is necessary. This proposal does not proscribe a specific prioritization strategy, but it is something important to consider as the during the actual implementation.

OPERATIONAL - DATABASE MANAGEMENT FUNCTIONS

- **ACCESS CONTROL** - For security purposes, access control allows an administrator to define policies specifying who can access the database and what those users can update the database.
- **OWNERSHIP** - The access control is also tied closely to the concept of ownership. The database is the steward of the data, but there will be some other ONAP component or function that is the owner of the data. The solution is tied to the access solution. What entity that has access and is the master of the information shall be incorporated into the database solution.
- **CONTROL INTERFACE GUI** - The database should have a means of control for the user or administrator, such that they can perform database operations and management functions. Such as clustering, backup & restore, and partitioning etc.
- **DATABASE LOGGING** - The database should provide a means for an operator to see a history of operations that has been performed tracking who did what operation when. The user should be able to view a history of operations that have been performed.
- **NETWORK AUDITING** - It will allow for auditing of configuration and parameters against a known quantity such as a "golden" template. It itself stores & provides the data for another thing to perform the auditing. There is a network use case of data record and what's in the network and insuring what has been designed in the model is enforced in the network.

OPERATIONAL - DATA INTEGRITY & RECOVERY

- **DATA RECOVERY** - It will allow for the recovery of data based on a source of truth. The source of truth would be defined by either by a user or predefined from a vendor specification.

- **BACK UP & EXPORT** - Often to facilitate data recovery, trouble shooting, and migration tasks, an export or backup of the database would be performed by a user. The backup & export are also database management functions that should be supported on the database interface. Backup can mean to save the contents of the database into a specified repository. Export could entail taking the contents of the database and sending it to a file which can be analyzed.
- **RESTORATION** - It will allow for the recovery and restoration of data to a fall back point or from a database backup.
- **ROLL BACK** - It will allow for rollback back to "save points" or recovery points. Note: Roll back is a shorter duration than restoration and is more resource intensive than a restoration.
- **DATA HISTORY** - It will allow an operator to see a history of changes in database. This will include the versioning of the data, which is also associated with updates, roll back, and restoration. Granularity of history. Hysteresis. Historical State (keep up with changes as they arrive). Race Conditions. Ripple effect of changes. Object relations to update. Tracking updates/deletions/changes. Consistency - in absolute sync with network.
- **QUALITY AUDITING** - It will allow for auditing of configuration and parameters against a known quantity such as a "golden" template. It itself stores & provides the data for another thing to perform the auditing. Auditing also entails consistency checks.

ACCESS - CONSUMER API

- **CONSUMER API** - There shall be a consumer interface which would allow an authorized system to access the database. An authorized system is any system, internal or external, that has been authorized to communicate with the RunTime Config DB. A consumer is defined as anything: human or machine, that is making use of the database.
- **API Type** - A RESTful API is preferred as it is commonly used within ONAP. However, other API types and communication mechanisms could also be considered as well.
- **CONNECTIVITY** - The consumer API should support consumers that are categorized as Northbound, Southbound or West/Eastbound.
- **SYNCHRONIZATION EXTERNAL DATABASES** - It may be possible that synchronization to other external customer data bases that a service provider may have

DEPENDENCIES

ONAP PROJECT DEPENDENCIES

A dependency something that RunTime Config DB needs to operate properly.

Dependencies vs Scope

DEPENDENCIES – need to operate

SDC Yang Model (to load schema)
ability to process & translate yang models into schemas
AAF (intra-ONAP security)
Database implementation for Data Persistency
(for example MariaDB)



SCOPE



RunTime Configure DB

RECEIVE INFORMATION
WRITE INFORMATION
PUBLISH CHANGES
REFERENTIAL INTEGRITY
INGEST PACKAGES
LOGICAL OBJECTS
ASSOCIATIONS
CARDINALITY RULES
LINKING RESTRICTIONS
SYNCHRONIZATION
DATA INTEGRITY & RECOVERY

DEPENDENCIES – value added

DMaaP (some use cases to work / indirect dependency)



Direct Dependencies - RunTime Config DB depends directly on the following ONAP projects:

- **AAF / Security** - basic intra-ONAP security
- **SDC (Design Time)** - Service CSAR package used for initial database schema definitions
 - Yang Model definition are produced SDC Design Studio
 - These Yang models are important to process and setup the database schema
- **MariaDB** - Database technology would be used to handle Persistency which allows for Data History management

Indirect Dependencies - The indirect dependencies are things that would enhance the operation of the RunTime Configuration Database, or that might be used in applications or use cases but are not absolutely necessary for basic operation:

- **DMaaP (DCAE)** - VES collection and DMaaP notifications. *Indirect dependency* because the RTCfgDB uses notifications off of DMaaP for example for CMNotify events.

OPEN SOURCE DEPENDENCIES

RunTime Config DB depends upon the following open source projects:

As the project becomes more mature, it will be determined what sort other dependent open source functions would be important for this project. Examples might include, MariaDB, a non-SQL database, or Casandra.

For temporal analysis, the enhancements to database would have other open source dependencies to manage temporal data. Some examples of this are Influx db, Posgress *TimeScale*.

RESOURCES

•Primary Contact Person:

- [Benjamin Cheung](#)
- [Tony Finnerty](#)
- [Toine Siebelink](#)

Table of Committers: (+2)

NAME	GERRIT ID	COMPANY	EMAIL	TIME ZONE
Toine Siebelink	Toine Siebelink	Ericsson	toine.siebelink@est.tech	UTC+1
Rishi Chail	Rishi Chail	Ericsson	rishi.chail@est.tech	UTC+1
Bruno Sakoto	Bruno Sakoto	Bell Canada	bruno.sakoto@bell.ca	UTC-5

Table of Contributors: (+1)

(Names and affiliations of any other contributors)

NAME	GERRIT ID	COMPANY	EMAIL	TIME ZONE
Benjamin Cheung	Benjamin Cheung	Nokia	ben.cheung@nokia.com	UTC-4
N. K. Shankar	N.K. Shankaranarayanan	AT&T	shankar@research.att.com	UTC-4
Marcin Krasowski	Marcin Sebastian Krasowski	Samsung	m.krasowski@samsung.com	UTC+2
Pawel Slowikowski	Pawel Slowikowski	Samsung	p.slowikows2@samsung.com	UTC+2
Krystian Kedron	user-7f92d	Samsung	k.kedron@partner.samsung.com	UTC+2
Carlos Manzanares	Carlos Manzanares	Nokia	Carlos.Manzanares@nokia.com	UTC+3
Fred Feisullin	Fred Feisullin	Verizon Wireless	Fred.Feisullin@Verizon.com	UTC-4
Sandeep Shah	Sandeep Shah	IBM	Sandeep.Shah@ibm.com	
Philippe Leger	Philippe Leger	Bell Canada	Philippe.Leger1@bell.ca	UTC-4
Joachim Blixt	Joachim Blixt (Samsung)	Samsung	j.blixt@partner.samsung.com	UTC+2
Tony Finnerty	Tony Finnerty	Ericsson	tony.finnerty@ericsson.com	UTC+1
Theodore Johnson	theodore johnson	AT&T	johnsont@research.att.com	UTC-4
Niamh Core	Niamh Core	Ericsson	niamh.core@est.tech	UTC+1
Aditya Puthuparambil	aditya puthuparambil	Bell Canada	aditya.puthuparambil@bell.ca	UTC+1
Claudio D. Gasparini	Claudio David Gasparini	Pantheon.tech	claudio.gasparini@pantheon.tech	UTC+1
Ruslan Kashapov	Ruslan Kashapov	Pantheon.tech	ruslan.kashapov@pantheon.tech	EET (UTC+2)
Martin Vezeau	Martin Vezeau	Bell Canada	martin.vezeau@bell.ca	UTC-4

Other Contact People:

- [Zu Qiang \(Ericsson\)](#)
- David Kinsey
- [KINGSLEY LAWRENCE](#)
- [Swaminathan Seetharaman](#)
- [N.K. Shankaranarayanan](#)
- Melanie Sater
- [Anita Hai](#)
- [Marge Hillis](#)
- [Marc-Alexandre Choquette](#)
- [Michela Bevilacqua](#)
- Mike Elliott
- [Munir Ahmad](#)
- [Junfeng Wang](#)
- [Bruno Sakoto](#)
- [Fred Feisullin](#)
- [Claudio David Gasparini](#)

KEY PROJECT FACTS

Project Name:

- JIRA project name: *cps*
- JIRA project prefix: *cps*

Repo name:

- *cps*

Lifecycle State: incubation (<https://wiki.onap.org/display/DW/ONAP+Project+and+Component+Lifecycle>)

Primary Contact: [Toine Siebelink](#) (PTL)

Project Lead: [Toine Siebelink](#) (PTL)

mailing list tag: *cps*

TSC APPROVAL

*Link to TSC approval:


Link to approval of additional submitters:


Link to TSC Meeting: [TSC 2020-10-01](#)

Item	Link / Recording
TSC Slides	ConfigurationPersistencySvcR8TSC_202010Oc01v11.pptx
	ConfigurationPersistencySvcR8TSC_202010Oc01v11.pdf

TSC Slides Updated	<div>  <p>ConfigurationPe...010Oc02v11.pptx</p> </div> <div>  <p>ConfigurationPe...2010Oc02v10.pdf</p> </div>
Video	202010Oc01_zoom_0.mp4
Audio	202010Oc01_audio_only.m4a


ASSOCIATED FILES

File	Description
R8 Honolulu CPS Overview Presentation PPTX	<div>  <p>ConfigurationPe...010Oc02v10.pptx</p> </div>

R8 Honolulu CPS Overview Presentation PDF	 ConfigurationPe...2010Oc02v10.pdf
R8 Presentation made at the Design Developer Forum (DDF) 15 Oct 2020	ConfigurationPersistenceSvcDDF_202010Oc15v13.pptx ConfigurationPersistenceSvcDDF_202010Oc15v13.pdf

USE CASES

As some initial examples, to help the reader understand the use case. Some of the first uses and use cases involving this project will be a Run-Time Configuration data persistency application applied to the OOF/SON PCI Release 6 Use Case.

Use Cases / PoC	Description
OOF SON PCI Use Case	<p>OOF SON PCI Use cases which is a 5G application to manage PCI values</p> <p>In R6: OOF (SON) in R5 El Alto, OOF (SON) in R6 Frankfurt</p> <p>In R7:</p>
End to End Network Slicing (R6)	<p>In 5G End to End Network Slicing Use Case to provide the management of Slices.</p> <p>C&PS will be used in the Network Slicing Use Case in R6 as one of its applications:</p> <p>In R6: NETWORK SLICING PoC in R6 Frankfurt (Obsolete)</p> <p>In R7: E2E Network Slicing Use Case in R7 Guilin</p>
Model Driven Proof of Concept (Ericsson)	<p>A proof of concept for model driven configuration management integrating standards driven parameters and introducing a</p> <div>  CPS POC Proposal.pptx </div> <p>platform for Network Element configuration management.</p> <p>Slides presented at the LFN DDF 25/06/20: DDF_CPS_PoC.pptx</p>

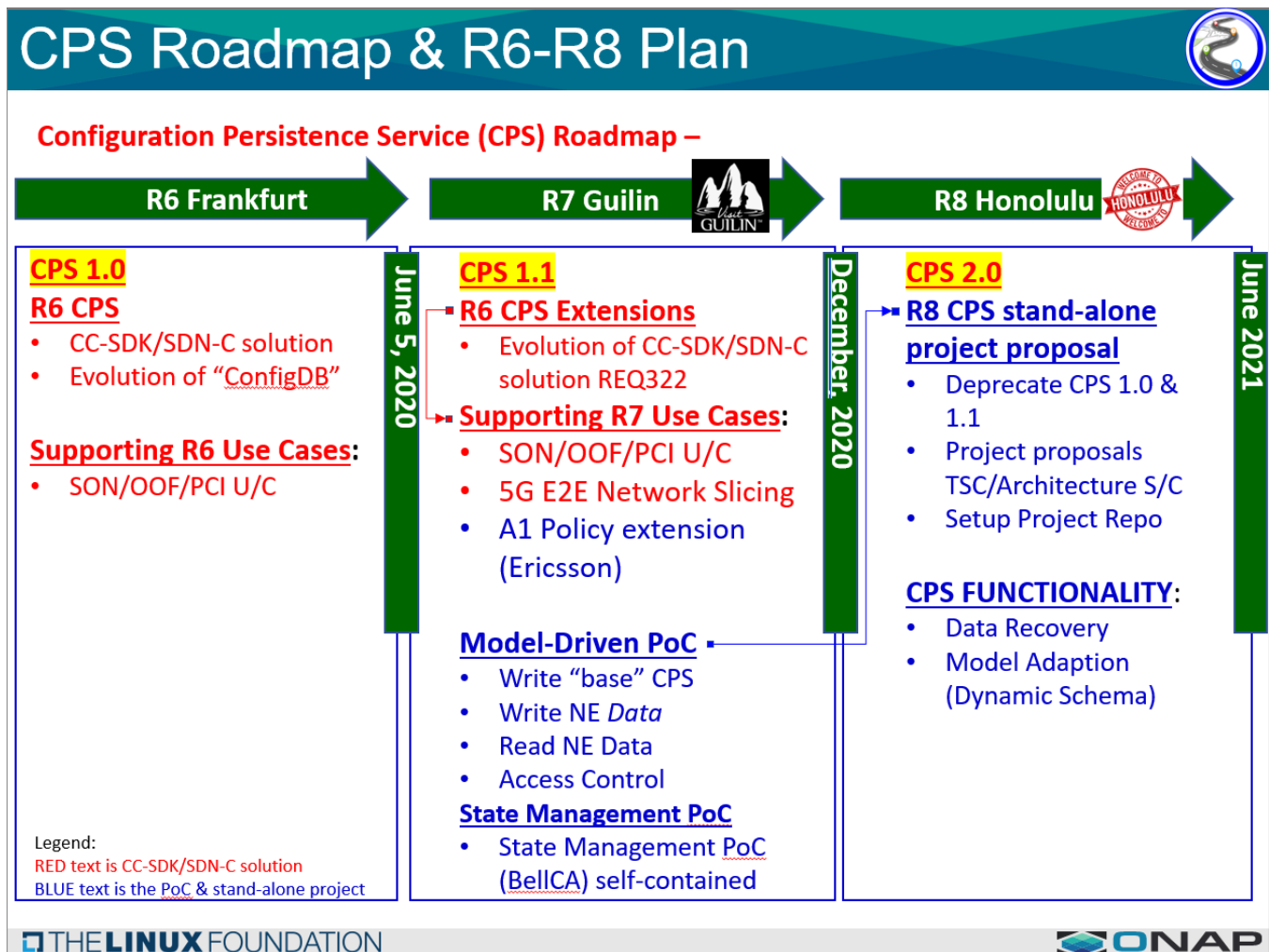
State Management Proof of Concept (Bell Canada)

Time series database and dynamic (model agnostic) supporting the configuration & operational of states. This proof of concept will focus on the management of states related to xNFs (any resources) in a network.

ROADMAP

The expected Roadmap & Delivery Plan from the initial releases is:

The C&PS Roadmap from R6 to R8

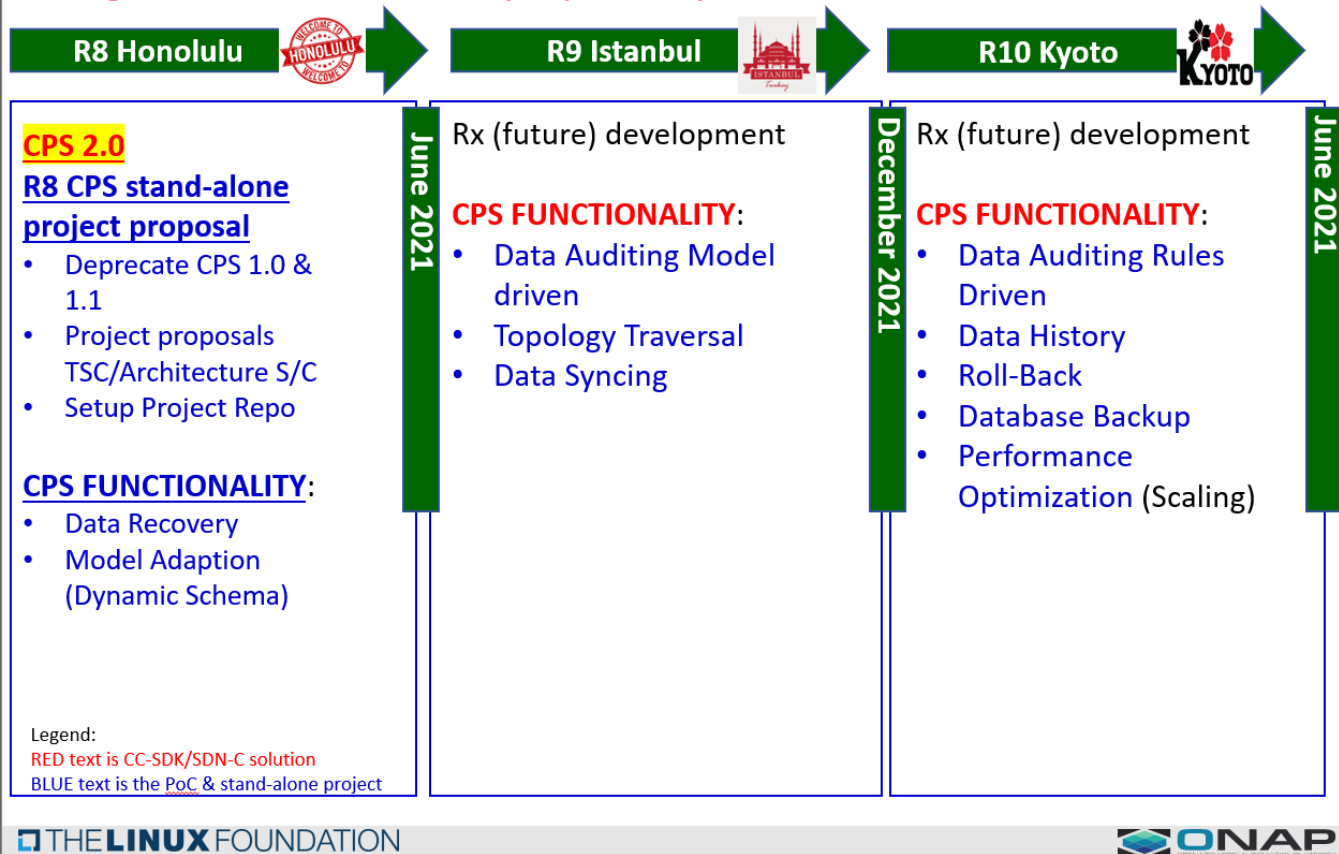


Continuing on ...

CPS Roadmap & R8-R10 Plan



Configuration Persistence Service (CPS) Roadmap –



The following is a table of the CPS release Pages:

Release	Description	Wiki Page
R6 Frankfurt	R6 CPS Use Case Project Page	CONFIGURATION PERSISTENCE SERVICE R6
R7 Guillin	R7 CPS Use Case Project Page	Configuration & Persistency Service R7
R8 Honolulu	R8 CPS Use Case Project Page	Configuration Persistence Service (CPS) for R8 Honolulu

NEXT STEPS (Incubation)

The following Next Steps are from the component lifecycle page:

[ONAP Project and Component Lifecycle](#)

The current view from the project team on these are:

Next Step	Description	Response
1	1A. Name of the project is appropriate (no trademark issues etc.);	Official Name of the project will be CONFIGURATION PERSISTENCE SERVICE
	1B. Proposed repository name is all lower-case without any special characters	Apache Geode is also using CPS, but it is not trademarked. Geode CPS

2	Project contact name, company and email are defined and documented	Toine Siebelink is the PTL and info can be found at KEYPROJECTFACTS
3	Description of the project goal and its purpose are defined	The goal and purpose are on the project proposal: PROJECTDESCRIPTION
4	Scope and project plan are well defined	The scope is defined in the project proposal: SCOPE
5	Resources committed and available (company commitments, all people)	These can be found here: RESOURCES Repo TBD
6	<i>Contributors</i> identified (people who add code) +2	These can be found here: RESOURCES
7	Initial list of <i>committer</i> identified (elected/proposed by initial contributors) +1	These can be found here: RESOURCES
8	Meets ONAP TSC Policies (NFR, GR, Code review, Testing, Audits, Documenting Interface)	The C&PS team intends to follow all ONAP TSC Policies .
9	Proposal has been socialized with potentially interested or affected projects (e.g. AAI, CDS, SO) and/or parties	TSC - TSC 2020-10-01 Requirement S/C - ONAP R8 Modeling High Level Requirements Architecture S/C - Modeling S/C - ONAP R7 Resource IM Call 2020-9-21 E2E Network Slicing (Requirements/Use Cases) SON PCI (Requirement/Use Cases)
10	Cross Project Dependencies (XPDs). In the case where a project will require changes in other projects, those projects are listed in the proposal, and a sponsoring developer in the project has been identified	There are no changes required upon other ONAP Platform component projects.
11	Tools have been identified and discussed with relevant partners (Linux Foundation, IT). Once the project pass the review: 1. the tools chain must be created within one week. 2. Tools encompass Configuration Management, CI-CD, 3. Code Review, 4. Testing, 5. Team Wiki, 6. End Users documentation (not exhaustive)	1. Tool Chain: (action) 2. Tools: (action) 3. Code Review: using Gerrit for Code Review. 4. Testing: unit, integration test will be part of the developer work; ONAP goal for 55% testing code coverage. Will be using SonarQube. 5. Project Page: Configuration & Persistency Service R7 Developer page: Configuration & Persistency Service PoC 6. Documentation: (RST files will be in the CPS repo)