Contribution and Development

Child pages:

Overview

This page is to provide detail on the contribution process for CPS.

Joining the ONAP Community

- Joining the ONAP Technical Community

Development Procedures and Policies

- Development Procedures and Policies

Code Repos

Component	ONAP Gerrit Repo	Notes
CPS-Core, NCMP	https://gerrit.onap.org/r/admin/repos/cps	
NCMP-DMI-Plugin	https://gerrit.onap.org/r/admin/repos/cps/ncmp-dmi-plugin	
Temporal DB	https://gerrit.onap.org/r/admin/repos/cps/cps-temporal	No longer maintained
TBDMT (mapper tool)	https://gerrit.onap.org/r/admin/repos/cps/cps-tbdmt	Hosted by CPS Team, maintained by Wipro

Code Contribution

- Configuring Gerrit

Kanban board

https://jira.onap.org/secure/RapidBoard.jspa?projectKey=CPS&rapidView=228

Antlr plugin set up for cps-path-parser (optional)

The cps-path-parser module uses Antlr. IntelliJ has an Antlr-v4 plugin If you want to use it it needs to be configured as follows:

1. Right-click on cps-path-parser`/src/main/antlr4/org/onap/cps/cpspath/parser/antlr4/CpsPath.g4` file in the project file tree.
2. Select "Configure ANTLR..."
3. Set "Output directory where all output is generated" to `<your git repo>\cps-path-parser\target\generated-sources\antlr4`
4. Set "Package/namespace for the generated code" to: `org.onap.cps.cpspath.parser.antlr4`
5. Click [OK]

If using Intellij, it may still return an error where it cannot find the generated sources for Antlr4.

To resolve this:

1. Right click on the module "cps-path-parser" in your cps project.
2. Click "Open Module Settings".
3. To the right, highlight where you set the output directory for generated sources
4. Click "Mark as: Sources"
5. Rebuild the project, and Intellij can now see the new generated sources.

CPS Specific Code Style set up

- **InteliJ IDE**: As we have noticed some slight formatting issues that are not controlled by the IDE we prefer that CPS contributors use IntelliJ (community edition is perfectly good for Java development)
- **CPS CheckStyle Scheme**: CPS as extended the standard ONAP CheckStyle configuration. Once you have downloaded CPS you can find the definition in *<your_git_folder>checkstyle/src/main/resources/cps-java-style.xml*
- **Groovy**: CPS uses Groovy/Spock for unit testing. There is some additional formatting setup to ensure consistent formatting of Groovy files.

Configure CPS code CheckStyle Plugin for IntelliJ

1. Select, File, Settings, Tools, Checkstyle
2. Click on + beside the 'Configuration File' box to add a configuration
3. Set description to something like 'ONAP Rules'
4. Click on Browse to select the file *<your_git_folder>checkstyle/src/main/resources/cps-java-style.xml*
5. Complete the Wizard (you can set exclusion properties if needed)
6. Activate the Configuration File you just added by selecting the relevant checkbox
7. optional: Click [OK] to close the settings popup

Configure CPS code style auto formatting for IntelliJ (using the same CheckStyle rules and automating it for you 🙂)

1. Select, File, Settings, Editor, Code Style
2. Click on the gear icon at the end of the line for "Scheme:"
3. Optional: As importing a schema overrides the current scheme you might want to first use the 'Duplicate..' and 'Rename...' options to create an easily identifiable scheme e.g. '**CPS Standard**'
4. Import SchemeCheckstyle Configuration
5. Click on Browse to select the file *<your_git_folder>checkstyle/src/main/resources/cps-java-style.xml*
6. optional: Click [OK] to close the settings popup

Configure CPS Java code style auto formatting for IntelliJ

1. Select, File, Settings, Editor, Code Style, **Java**
2. Ensure the same scheme is selected (as suggested in step 3 in the previous instructions: 'CPS Standard')
3. Click [OK] to close the settings popup

Configure CPS Groovy code style auto formatting for IntelliJ

1. Select, File, Settings, Editor, Code Style, **Groovy**
2. Ensure the same scheme is selected (as suggested in step 3 in the previous instructions: 'CPS Standard') However does does not affect Groovy setting and you manually need to set below too:
    a. Select 'Tabs and Indents' tab
    b. Set 'Tab size' to **4**
    c. Set 'Indent' to **4**
    d. Set 'Continuation indent' to **4**
    e. Set 'Label indent' to **4** (this is the only one that is different from Java and it to indent under the give: when then: labels!)
    f. Select 'Imports' tab
    g. Set 'Class count to use import with '*'' to 999
    h. Set 'Names count to use static import with '*'' to 999
3. Click [OK] to close the settings popup

Code Submissions

Code Quality

- See CPS Quality
- CPS Code Review Check List

Commit Messages

- Commit message must be in the following format:

```
Comment explaining what is the purpose of the code.

Issue-ID: CPS-1
```

See also: Commit Messages

License Declarations

The following license template needs to be added and kept updated in CPS files (using the commenting scheme corresponding to each type of file):

License Template

```
/*
 * ============LICENSE_START=======================================================
 * Copyright (C) YYYY[-XXXX] <an organization>
 * Modifications Copyright (C) YYYY[-XXXX] <another organization>
 * ================================================================================
 * 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.
 *
 * SPDX-License-Identifier: Apache-2.0
 * ============LICENSE_END=========================================================
 */
```

Notes:

- SPDX-License-Identifier is required by Nordix and agreed by Bell Canada.

References:

- Copyrights and License Declarations
- Rules for implementing FOSS in a project
- https://wiki.nordix.org/display/DEV/Copyright+and+Licences+for+ONAP
- https://spdx.dev/ids/

Common Style Conventions

*This table just list the most commonly required style conventions in CPS, it is not to be intended to be complete

Language / Domain	Type of object	Style	Example
Java / Groovy	Package	lowercase	`org.onap.cps.rest.controller`
	Class	UpperCamelCase	`AdminRestController`
	Variable	lowerCamelCase	`cpsAdminService`
	Method	lowerCamelCase	`createDataspace`
JSON (incl event payload)	Property	lowerCamelCase	`cmHandles`
Open API / REST see ONAP Swagger Style	Resource / Node	lowercase(?) plural	`dataspaces`

	Model / Component	UpperCamelCase	`RestDmiPluginRegistration`
	Property	Conform to model	See below caveat*
SQL	Table	UPPER_SNAKE_CASE (singular)	`FRAGMENT`
	Field	UPPER_SNAKE_CASE	`DATASPACE_ID`
Liquibase	Table	lower_snake_case	`fragment`
	Field	lower_snake_case	`dataspace_id`
CNCF Header	header field	lowercase	`datacontenttype` `correlationid`

*A caveat to these styling conventions is that we should try to conform to the model which we are developing for. So if we model in yang we should manipulate that data using the kebab-casing convention as that is the convention used in yang models. However there are some endpoints where a convention has been set to use camelCasing when manipulating yang data. In these cases the current convention should be upheld to avoid backward incompatible changes.

Logging Guidelines

see CPS Logging Guidelines

Code Analysis

We recommend that most user stories are accompanied with small wiki page to document/clarify some analysis for that user story.

See Implementation Proposals

Unit Test

CPS is promoting and using Groovy and Spock for all our unit test. All new modified code should include good quality test.

If you are new to Groovy and Spock look at OneSummit Groovy & Spock Workshop Nov 2022 it includes a demo project with exercises and solutions

see Groovy & Spock you can also contact the PTL Toine Siebelink for more details

DB Schema Changes (Liquibase Change Sets)

See

User Story Demos

As part of our best practice to finish a user story it should be demonstrated to team and if possible recorded

see CPS User Story Demos

Jenkins Job

ONAP uses Jenkins based CICD tool chain. However, contributors are only given read access to the Jenkins servers. All jobs are created by automatic generation from JJB definitions.

https://jenkins.onap.org./view/cps/

Bug reporting

See Bug Reporting for details on reporting issues. Issues may be created here:

https://jira.onap.org/secure/RapidBoard.jspa?rapidView=228&projectKey=CPS&view=planning&issueLimit=100

Vulnerability report process

If you find any vulnerability please email the security contact with the information that you have.

The security contact is ; toine.siebelink@est.tech

We will give credit to anyone who reports a vulnerability so that we can fix it. If you wish to remain anonymous instead, please let us know and we will respect that.

Documentation

- Published (upon merge) here: CPS Documentation (
- All CPS documentation sources (.rst files) are included in the \docs folder of the CPS repo
- for linting & testing documentation changes locally follow: https://docs.onap.org/en/latest/guides/onap-developer/how-to-use-docs/setting-up.html#testing