

Northbound Interface Adapter Customization Development Guide for CLI-Based Protocol Ericsson Dynamic Activation 1

USER GUIDE

Copyright

© Ericsson AB 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing.

Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	Purpose and Scope	1
1.2	Target Group	1
1.3	Typographic Conventions	1
1.4	Prerequisites	1
2	Tutorial Project Overview	2
2.1	Development Process of Adapted Northbound SSH CAI Interface	3
3	Preparing for NBIA Development Environment	3
3.1	IDE Installation	3
4	Customized NBIA Submodule Development	4
4.1	Importing the SSH CAI NBIA Project to Eclipse	4
4.2	SSH CAI Northbound Interface Implementation	5
4.2.1	Project Structure	5
4.2.2	Northbound Interface Implementation	9
4.2.3	Provisioning Services Implementation	10
4.3	Packaging Northbound Adapter	11
5	Deployment	11
5.1	Deploying the Customized NBIA Submodule	12
5.1.1	Installing the Customized NBIA Submodule	12
5.1.2	Activating the Customized NBIA Submodule	12
5.1.3	Checking the Submodule Status	12
5.1.4	Adjusting Load Balance Configuration for SSH CAI Submodule	13
5.1.5	Uninstalling the Customized NBIA Submodule	16
5.1.6	Updating the Customized NBIA Submodule	16
5.2	Setting Properties for the Customized NBIA Submodule	17
5.2.1	Properties for the SSH CAI Submodule	18
5.2.2	Adding/Changing Customized Properties for the Customized NBIA Submodule	19
5.2.3	Modifying Properties Values for the Customized NBIA Submodule	20
5.2.4	Restore the Default Value of the Property for the Customized NBIA Submodule	21
5.2.5	Key Store Configuration for the SSH CAI Submodule	21
5.3	Configuring Logs	22



6	Verifying the Northbound Adapter	23
6.1	Creating a User	23
6.2	Login Through SSH Protocol	23
6.3	Sending a CAI Request	24
6.4	Error Code	24
	Reference List	27



1 Introduction

This document provides detailed instructions on how to develop, deploy, and verify customized Northbound Interface in the Northbound Adapter (NBIA) framework in Ericsson™ Dynamic Activation (EDA).

1.1 Purpose and Scope

This document is intended for the Customer Adaptation (CA) developers who develop the Northbound Interface with the CLI-based protocol in NBIA, targeting provisioning integration with customer's Business Support System Northbound Interface.

1.2 Target Group

The target group for this document is as follows:

- System integrators
- Application designers

1.3 Typographic Conventions

Typographic conventions are described in the document *Library Overview*, Reference [1].

1.4 Prerequisites

This document is written with the assumption that the users:

- Have good knowledge of Dynamic Activation, Java™ language, Eclipse, and OSGi/Karaf framework.
- Have read the following documents:
 - *Customization - Architectural Overview*, Reference [2]
 - *Northbound Interface Adapter Reference Manual*, Reference [3]

2 Tutorial Project Overview

The tutorial project offers an Eclipse project example that implements a customized northbound “CAI interface adapter over SSH protocol”. For more information about CAI interface, refer to *CAI Interface Specification for HLR Components*, Reference [5]. Unlike traditional CAI login procedure, this tutorial project applies SSH public-key authentication method for user authentication. For more information about SSH public-key authentication, see Reference [11].

The CAI interface adapter is the submodule under the NBIA module, including SSH protocol configuration and business interface related logic. The following figure shows the function design of this adapter:

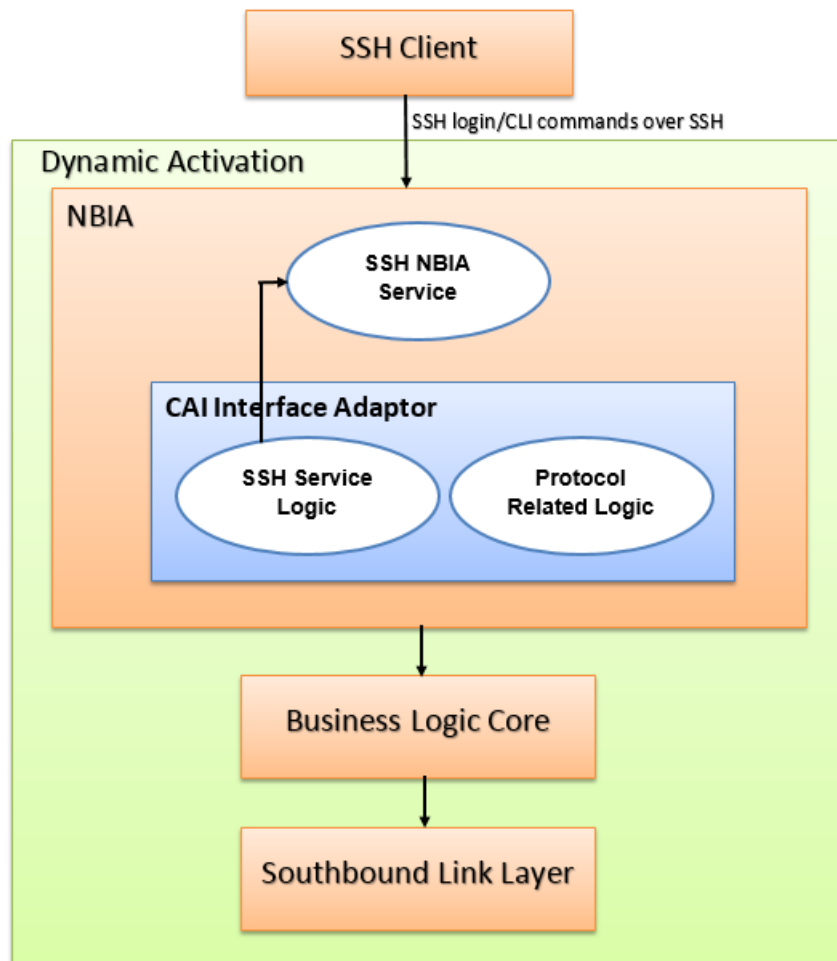


Figure 1 Northbound Interface Adapter Development Workflow



2.1 Development Process of Adapted Northbound SSH CAI Interface

The following figure shows the workflow of the development process:

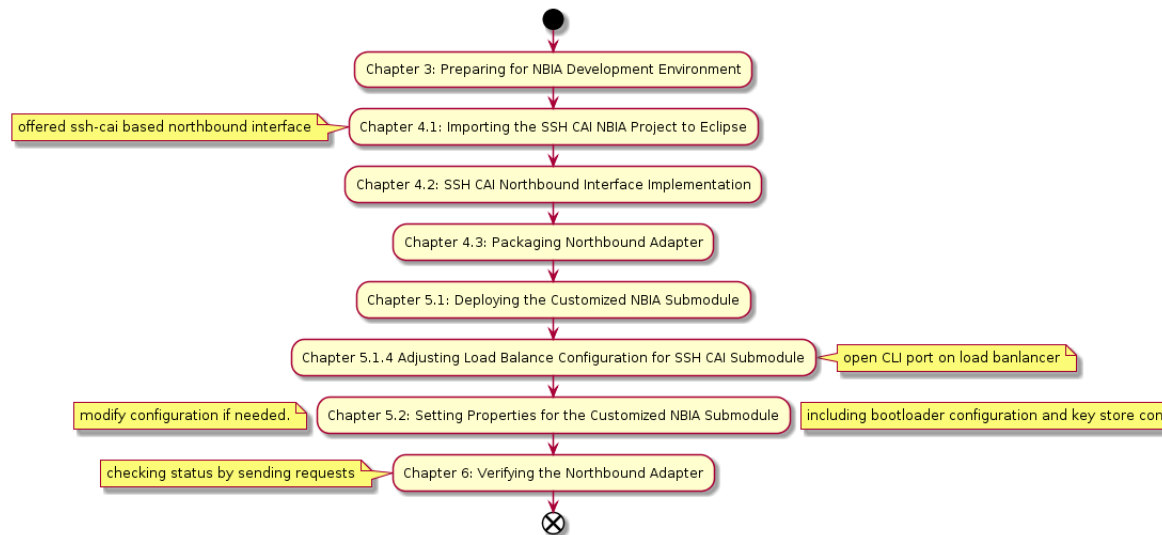


Figure 2 Development Workflow

3 Preparing for NBIA Development Environment

3.1 IDE Installation

Northbound Interface Adapter (NBIA) module is written in Java. It is recommended to develop NBIA module with Eclipse Integrated Development Environment (IDE) and build them in Maven.

For how to prepare JDK, Maven, and Eclipse environment, refer to *Customer Adaptation Development Guide for Resource Activation*, Reference [4].



4 Customized NBIA Submodule Development

Create a customized interface according to the steps as follows:

1. Use offered SSH protocol-based CAI interface as an example and import the source code into Eclipse.
2. Implement the business interface logics to handle customer-specific provisioning requests.
3. Package the code.

A customized northbound interface mainly covers the following two aspects:

- Customized northbound protocol implementation

In this document, the example project offers SSH protocol implementation as a reference.

- Customer-specific business interface logics implementation

In this document, the example project offers CAI logics implementation as a reference.

The following subchapters provide detailed information about CAI interface adapter projects and how to customize it regarding to specific customer needs.

4.1 Importing the SSH CAI NBIA Project to Eclipse

The example project `mpe-business-nbi-ssh-cai` is a design base project in this section. To find the example project, follow these steps:

1. Download the CA package `EDA_SDK_CUSTOMER_ADAPTION_SW-<version>.tar.gz` from SW Gateway in the Dynamic Activation distribution.
2. Extract the above CA package to get the subpackage `ca-sourcecode-<version>.tar.gz`.
3. Extract the preceding subpackage and find the project `mpe-business-nbi-ssh-cai` in the directory `Tools/NBIA/mpe-business-nbi-ssh-cai`.

Import `mpe-business-nbi-ssh-cai` to your workspace folder. The project is imported as existing Maven project.



4.2 SSH CAI Northbound Interface Implementation

4.2.1 Project Structure

The following figure shows the main project directory and configuration files:



```

└─ mpe-business-nbi-ssh-cai [prov-ca1 master]
  └─ src/test/java
  └─ src/test/resources
  └─ src/main/java
    └─ com.ericsson.mpe.business.nbi.ssh.cai
      └─ NbiaSshCaiImpl.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.cmd
        └─ Command.java
        └─ CommandNotSupported.java
        └─ CommandNotSupportedException.java
        └─ CommandProcesser.java
        └─ Get.java
        └─ OtherCommand.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.configuration
        └─ CAIConfiguration.java
        └─ CAIConfigurationValidator.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.handler
        └─ CAIRequestHandler.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.ssh
        └─ CaiSshCommand.java
        └─ CaiSshPublicKeyAuthenticator.java
        └─ CaiSshServer.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.util
        └─ DefaultEntityResolverImpl.java
        └─ DefaultErrorHandlerImpl.java
        └─ Log4JConfigurer.java
        └─ NamespaceList.java
        └─ PropertyUtil.java
        └─ SslContextFactory.java
        └─ XMLUtil.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.util.cai
        └─ CaiConstant.java
        └─ CaiFaultCodes.java
        └─ CaiStringTokenizer.java
        └─ CaiUtil.java
      └─ com.ericsson.mpe.business.nbi.ssh.cai.util.fault
        └─ ErrorCodes.java
        └─ ErrorMapping.java
    └─ src/main/resources
    └─ Referenced Libraries
    └─ JRE System Library [JavaSE-1.8]
  └─ config
    └─ log4j.dtd
    └─ mpe-business-nbi-ssh-cai-configurations.properties
    └─ mpe-business-nbi-ssh-cai-log4j.xml
    └─ mpe-business-nbi-ssh-cai-module.conf
    └─ nbia_ssh_keystore.jks
  └─ deploy
  └─ src
  └─ target
    └─ .classpath
    └─ .project
    └─ assembly.xml
```



4.2.1.1

Source Code Packages

The source code of this project includes the packages described in the following table:

Table 1 Source Code Packages

Java Package	Java Class	Description
com.ericsson.mpe.business.nbi.ssh.cai Implements the SSH CAI Northbound Interface. It includes submodule start and stop functions. For details, refer to Section 4.2.2 on page 9	NbiaSshCaiImpl	The class is OSGi bundle implementation class. It is configured in the OSGi configuration file <code>serviceComponentsInstance.xml</code>
com.ericsson.mpe.business.nbi.ssh.cai.configuration The package is CAI relative. If protocol is changed, the package is not applicable.	CAIConfiguration	Implements the CAI-related configuration logic
	CAIConfigurationValidator	Performs validation for some CAI configuration, for example, CAI port number
com.ericsson.mpe.business.nbi.ssh.cai.handler	CAIRequestHandler	Processes the input CAI request, performs simple validation for input string format, and passes the request to command processor.
com.ericsson.mpe.business.nbi.ssh.cai.ssh	CaiSshCommand	Extends the <code>AbstractNbiaSshCommand</code> class which implements the <code>Command</code> interface and transfers the CAI command to the <code>CAIRequestHandler</code> class. If protocol is changed, the class must be changed accordingly.
	CaiSshServer	Implements the <code>AbstractNbiaSshServer</code> class to get SSH service.



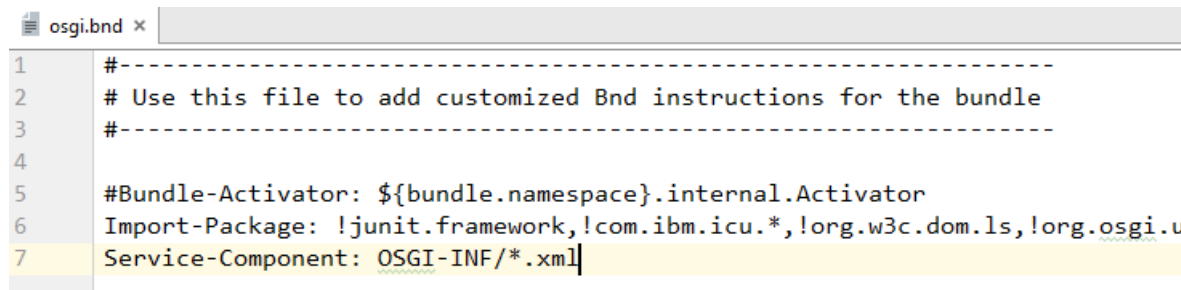
Java Package	Java Class	Description
com.ericsson.mpe.business.nbi.ssh.cai.cmd Implements CAI specific business logics. To apply for different customized CLI business logics, the developer must create their own specific CLI handling class instead. For details, refer to Section 4.2.3 on page 10.	CaiStringTokenizer	Extends the <code>StringTokenizer</code> class to handle CAI string tokens.
	Get	Extends the <code>Command</code> class. The <code>Get</code> class handles CAI <code>Get</code> command and processes proclog for the <code>Get</code> operation.
	OperationNotSupportedException	Extends the <code>Exception</code> class.
	OtherCommand	Extends the <code>State</code> class. It handles CAI command except for the <code>Get</code> operation and processes proclog for the command.
	Command	The parent class of the <code>Get</code> class and <code>OtherCommand</code> class. It offers services getting from <code>NbiaSshCaiImpl</code> , including alarm service, MBean service, OSGi service, and proclog service. And it offers some CAI-request-handling functions.
	CommandNotSupported	Handles invalid commands
	CommandProcessor	Processes a CAI command and changes the state accordingly. It sets the state in the session information map.
com.ericsson.mpe.business.nbi.ssh.cai.util	DefaultEntityResolverImpl	Implements <code>EntityResolver</code> . The class is intended to be used when the XML document uses either a DTD or schema that is not available.
	DefaultErrorHandlerImpl	Implements <code>ErrorHandler</code> . The error handler is used by the XML parser to report any errors that have occurred during parsing.
	Log4JConfigurer	Used to configure <code>log4j</code> from a settings file.
	NamespaceList	Implements <code>NamespaceContext</code> and <code>Serializable</code> . The class carries information of used namespace prefixes and URIs.
	PropertyUtil	Defines functions operating properties.
	SslContextFactory	Stores SSL certification.
	XMLUtil	Defines functions operating XML.
com.ericsson.mpe.business.nbi.ssh.cai.util.cai	CaiConstant	Defines constants used for CAI.
	CaiFaultCodes	Gives out all current allowed faults in CAI
	CaiUtil	Defines functions used by CAI process.
com.ericsson.mpe.business.nbi.ssh.cai.util.fault	ErrorCodes	Defines error codes.
	ErrorMapping	Maps errors.



4.2.1.2 OSGi Bundle Configuration

In some scenarios, for example introducing new 3PP into customized submodel, the OSGi bundle configuration (such as "import-package") must be updated. The following figure shows the OSGi bundle configuration files in the project. For more information, refer to official OSGi and Karaf documents.

The file `osgi.bnd` defines bundle activator and the imported packages used by the bundle.



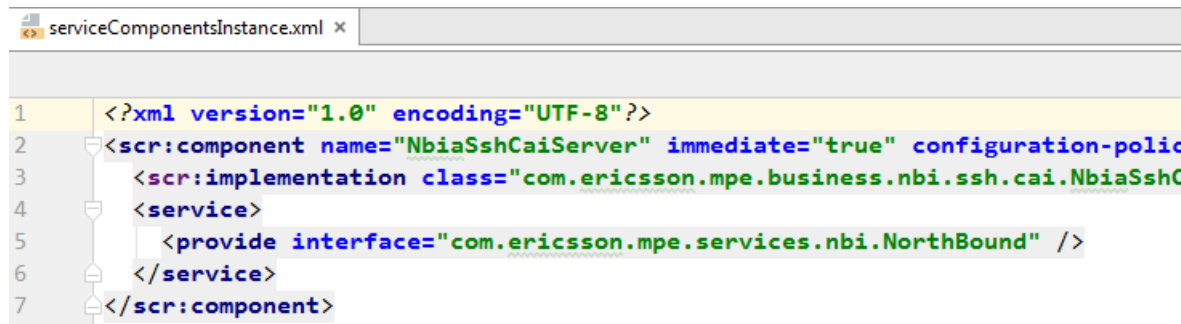
```

1  #-----
2  # Use this file to add customized Bnd instructions for the bundle
3  #-----
4
5  #Bundle-Activator: ${bundle.namespace}.internal.Activator
6  Import-Package: !junit.framework,!com.ibm.icu.*,!org.w3c.dom.ls,!org.osgi.u
7  Service-Component: OSGI-INF/*.xml

```

Figure 4 `osgi.bnd` File

The file `OSGI/serviceComponentsInstance.xml` defines the bundle component, including component name, implementation class, and interface. When doing customization, change the component name and implementation class accordingly.



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <scr:component name="NbiaSshCaiServer" immediate="true" configuration-policy="allow" >
3    <scr:implementation class="com.ericsson.mpe.business.nbi.ssh.cai.NbiaSshCaiImpl" >
4      <service>
5        <provide interface="com.ericsson.mpe.services.nbi.NorthBound" />
6      </service>
7    </scr:component>

```

Figure 5 OSGi Customization

4.2.1.3 Bootloader Configuration

The project also depends on several bootloader configurations. The default value of the parameters can be changed. For details, refer to Section 5.2 on page 17.

4.2.2 Northbound Interface Implementation

Class `NbiaSshCaiImpl` implements the Northbound Interface and is managed with dependency injection. NBIA provides some basic services by the

NBContext class, including alarm, authorization, configuration, processing logging, provisioning, and configuration.

For information about how to get all the preceding services by NBContext, refer to the start method of AbstractSshNorthBound, as shown in the following figure.

```
public void start(NBContext context) throws NBIException {
    LOGGER.info("AbstractSshNorthBound start ...");
    setAlarmService(context.getAlarmService());
    setAuthenticationService(context.getAuthenticationService());
    setMBeanService(context.getMBeanService());
    setExtensionService(context.getExtensionService());
    setProcLogService(context.getProcLogService());
    setProvisionService(context.getProvisionService());
    if(this.sshServer != null) {
        try {
            LOGGER.debug("Start SSH server {}. ", this.componentName);
            this.sshServer.start();
        } catch (Exception var3) {
            LOGGER.error("Failed to start {}. ", this.componentName, var3);
            throw new NBIException(General.SERVICE_UNAVAILABLE.getCode(), "Failed to start the SSH se
        }
    }
}
```

Figure 6 Start Method of AbstractSshNorthBound

In the current example implementation, the services are retrieved from the Command java class.

4.2.3 Provisioning Services Implementation

In this example, CAI interface is implemented to handle CAI requests, including the Create, Set, Get, and Delete operations.

The processing logic is implemented in the com.ericsson.mpe.business.nbi.ssh.cai.cmd package.

The main processing logic includes:

- **Execute command:** The main processing logic is to map a CAI request to an internal Northbound Interface request. Then refer to the CommandProcessor.processCommand, Command.stateEnter, Get.getResponse, and OtherCommand.getResponse methods.
- **Build a Northbound Interface request, by referring to Get.prepareRequest and OtherCommand.prepareRequest.** Assign the following attributes and generate a Northbound Interface request:
 - **SessionData:** A set of session-related data, such as username, proclog ID, and session ID. SessionData carries the session-specific data, which is used by the Dynamic Activation framework.
 - **Operation:** Assign operation attributes according to the request type (Create, Get, Set, and Delete).



- **MO_NAME**: It is the logical name of the target MO. In CAI protocol, the value of **MO_NAME** is as same as the one of MO type. Customers can set their own **MO_NAME** only if NE can parse it. **MO_NAME** is used to configure the routing on Dynamic Activation GUI.

For example, the **MO_NAME** of CAI protocol: AUCSUB
- **RawRequest**: The original request sent from the Northbound Interface.
- **MoAttributes**: It is an XML document that contains all required information for the BL. The **MoAttributes** is used to transfer data from the Northbound Interface to proxy, and it is transparent to the current Dynamic Activation service, because it contains BL-related information.
- **Parse the Dynamic Activation response**: Parse the Northbound Interface response and map its attribute to corresponding CAI response attributes. For details, refer to the `Get.buildResponseString`, and `OtherCommand.buildResponseString` method.

4.3 Packaging Northbound Adapter

Use Maven to make the `tar.gz` package of customized NBIA submodule. Confirm the Maven configuration in the `assembly.xml` file, which is like the following one:

```
<assembly>
  <id>sources</id>
  <formats>
    <format>tar.gz</format>
  </formats>
  <includeBaseDirectory>>false</includeBaseDirectory>
```

Figure 7 `assembly.xml` File

Build the `tar.gz` package by right-clicking the file and selecting **Run As > Maven install** in Eclipse IDE.

5 Deployment

After completing the development of customized NBIA submodule, a `tar.gz` package is ready for deployment. Place the `tar.gz` file under the directory `/home/bootloader/CArepository`.



Note: The system controller node referred in this section are:

- The SC node in native deployment
- Node-1 in virtual and cloud deployment

5.1 Deploying the Customized NBIA Submodule

5.1.1 Installing the Customized NBIA Submodule

Run the following command from a system controller node to install a customized NBIA submodule:

```
# bootloader.py submodule add --name/-n <submodule tar file>
--type/-t nbi --parent/-p nbia-module --host <PL hostname>
```

Note: <PL hostname> is the hostname of PL nodes. In virtual environment, <PL hostname> is the hostname of each node.

For example:

```
# bootloader.py submodule add -n mpe-business-nbi-ssh-cai-
4.27.tar.gz -t nbi -p nbia-module --host EBS-PL-3
```

Note: Restricted by OSGi Karaf framework, bundle version must be 4.xx and not higher than parent module version. For example, if nbia-module is 4.28, then mpe-business-nbi-ssh-cai version deployment fails if the version is 4.29 or 3.xx.

5.1.2 Activating the Customized NBIA Submodule

Run the following command from a system controller node to activate the customized NBIA submodule:

```
# bootloader.py node activate --host <PL hostname>
```

For example:

```
# bootloader.py node activate --host EBS-PL-3
```

5.1.3 Checking the Submodule Status

Run the following command from a system controller node to check the submodule version and status of the NBIA module and customized NBIA submodule:

```
# bootloader.py node status --host <hostname>
```

For example:



```
# bootloader.py node status --host all
```

The status of nbia-module must be Running as follows.

Host	Module	Version	Status	Bindings
node1	nbia-module	4.27	Running	OK(2/2)
node1	-> mpe-business-nbi-ssh-cai	4.27		

5.1.4 Adjusting Load Balance Configuration for SSH CAI Submodule

For submodule SSH CAI, SSH port must be added in load balance configuration.

5.1.4.1 Add Port for Virtual or Cloud Environment

Note: In the following step, @NBIA_SSH_CAI_PORT@ is the SSH port configured for the NBIA_SSH_CAI_PORT parameter. For details, refer to Section 5.2.1 on page 18. The default value is 3322.

To add SSH port for virtual or cloud environment:

1. On node-1, edit the Haproxy configuration file.

Run the following command as the root user:

```
# vi /etc/puppet/modules/haproxy/templates/haproxy_cfg.
erb
```

Add the port to be used. An example output is as follows:

```
listen cai3gIPv4_<% if @vip_external_trf_check
%><%= 'vip_trf' %><% else %><%= 'vip_oam' %><% end
%>:@NBIA_SSH_CAI_PORT@

bind <% if @vip_external_trf_check %><%= @vip_exte
rnal_trf %><% else %><%= @vip_external_om %><% end
%>:@NBIA_SSH_CAI_PORT@ transparent

mode tcp

balance leastconn

<% @list_of_nodes_for_haproxy.split(',').each do |ip|
-%> server <%= ip %>:@NBIA_SSH_CAI_PORT@ check

<% end %>
```

2. Wait for about 30 seconds. Check if the configuration is updated by checking the file /etc/haproxy/haproxy.cfg. Check if the configuration is loaded by using Haproxy dameon.



3. Use a web browser and log into the HAProxy Stats GUI through the following address:

```
http://<VIP-OAM-IP>:9000/haproxy_stats
```

Username: **admin**

Password: **Ericsson1**

4. Edit the file `/etc/puppet/modules/firewall/files/config/iptables-pg-product-rules.cfg` and insert the following lines:

```
-A input_ext -p tcp -m tcp --dport @NBIA_SSH_CAI_PORT@
-j ACCEPT
```

5. Save the file.
6. Wait for about 30 seconds and check the active rules.

For more information on how to check the active rules, refer to *System Administrators Guide for Virtual and Cloud Deployment*, Reference [6].

5.1.4.2 Add Port for Native Environment

To add SSH port for native environment, log on to the first SC node, SC1, and run the following commands:

Note: The following is an example about how to add a new provisioning port “3322” on EBS cluster. The port is the one configured in `NBIA_SSH_CAI_PORT`. For details, refer to Section 5.2.1 on page 18.

IP 10.175.160.225 is the traffic IP of eVIP. Replace it with corresponding one.

```
# telnet ` /opt/vip/bin/getactivecontrol` 25190
```

```
EVIP> enable
```

```
OK
```

```
EVIP# configure terminal
```

```
OK
```

```
EVIP(config)# alb alb_1
```

```
OK
```

```
EVIP(config-alb_1)# add flow-policy NBIA-submodule-interface => here to add a new flow policy
```

```
OK
```



```
EVIP(config-alb_1-NBIA-submodule-interface)# set
address-family ipv4 => begin to set parameter for this
flow policy
```

OK

```
EVIP(config-alb_1-NBIA-submodule-interface)# set protocol
tcp
```

OK

```
EVIP(config-alb_1-NBIA-submodule-interface)# set
target-pool PL_trf
```

OK

```
EVIP(config-alb_1-NBIA-submodule-interface)# set dest
10.175.160.225
```

OK

```
EVIP(config-alb_1-NBIA-submodule-interface)# set dest-port
3322
```

OK

```
EVIP(config-alb_1-NBIA-submodule-interface)# commit
```

OK

```
13. EVIP(config-alb_1)# show flow-policies
```

Flowpolicies:

8a: NBIA-submodule-interface (0x22c65b0)

99: PL-TRF (0x920ca0)

233: PL-aTRF (0x920dd0)

23a: PL-sTRF (0x920f10)

24d: PL-edifact (0x91d850)

270: PL-TestTRF (0x921050)

2dd: PL-cai2 (0x920a10)

2de: PL-cai1 (0x8ef8c0)

308: PL-mml (0x91d380)

346: PL-sTestTRF (0x921190)

Flowpolicy tree: (1 target pools)

Target pool: PL_trf (1 vips)

VIP 10.175.160.225: (10 ports)

Dport 3010: (1 protocols)

Protocol tcp: (1 srcs)

Src 0.0.0.0/0: (1 ports)

Src port 0 (PL-edifact)

so_group 0 :

...



```

Dport 3322: (1 protocols)
  Protocol tcp: (1 srcs)
    Src 0.0.0.0/0: (1 ports)
      Src port 0 (NBIA-submodule-interface)
        so_group 0 :
OK

EVIP(config-alb_1)# exit

OK

EVIP(config)# exit

OK

EVIP# save-config

OK

```

For more information about adding port from IPtables, refer to section Custom IPtables Rules - Native Deployment Only in *Security and Privacy Management*, Reference [8].

5.1.5 Uninstalling the Customized NBIA Submodule

Run the following command from a system controller node to uninstall the customized NBIA submodule:

```
# bootloader.py submodule delete --name/-n <submodule name>
--parent/-p nbia-module --host <PL hostname>
```

Note: If it is virtual env, <PL hostname> is the hostname of each node.

For example:

```
# bootloader.py submodule delete -n mpe-business-nbi-ssh-cai
-p nbia-module --host EBS4-PL-3
```

After uninstalling the submodule, the submodule must be removed from the module list. For how to check module status, refer to Section 5.1.3 on page 12.

5.1.6 Updating the Customized NBIA Submodule

Follow these step to update a customized NBIA submodule:

1. From a system controller node, run the following command to update the customized NBIA submodule:



```
# bootloader.py submodule update --name/-n <submodule tar
file> --type/-t nbi --parent/-p nbia-module --host <PL
hostname>
```

<PL hostname> is the hostname of the PL node to which the customized NBIA submodule is updated.

For example, update mpe-business-nbi-ssh-cai customized NBIA submodule on PL-3:

```
# bootloader.py submodule update -n mpe-business-nbi-ss
h-cai-4.27.tar.gz -t nbi -p nbia-module --host EBS-PL-3
```

2. From a system controller node, run the following command to activate the PL node:

```
#bootloader.py node activate --host <PL hostname>
```

<PL hostname> is the hostname of the PL node to which the customized NBIA submodule is updated.

For example:

```
# bootloader.py node activate --host PL-3
```

5.2 Setting Properties for the Customized NBIA Submodule

After importing the example submodule in Dynamic Activation IDE, the following configuration files are imported for setting the user properties:

- mpe-business-nbi-ssh-cai-configurations.properties – This file contains a list of all the configurable parameters.
- mpe-business-nbi-ssh-cai-module.conf – This file contains the setting of the default values of the parameters that are loaded by bootloader.
- mpe-business-nbi-ssh-cai-log4j.xml – This file is used for processing log configuration.
- nbia_ssh_keystore.jks – The default key store file

The following submodule is used in this section to describe the process of setting properties.

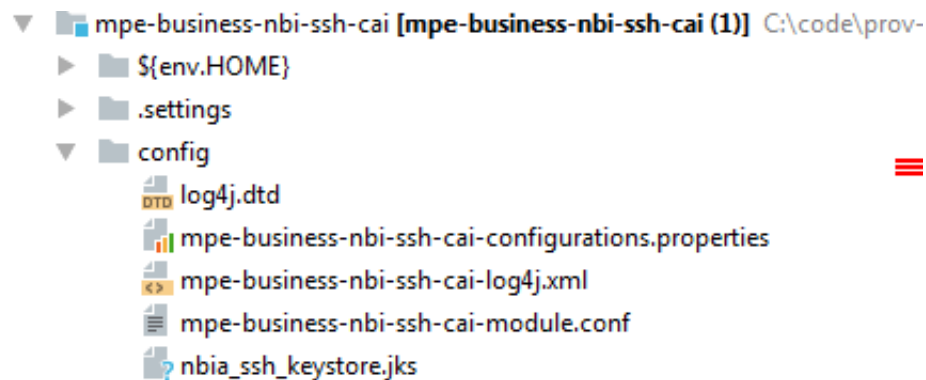


Figure 8 mpe-business-nbi-ssh-cai Submodule

5.2.1 Properties for the SSH CAI Submodule

The SSH CAI submodule offers several `bootloader` configuration attribute parameters. All the configurations can be changed using the `bootloader` tool. Refer to Section 5.2.5 on page 21 for how to change the attributes.

Table 2 Configuration Attributes

Configuration Attribute Name	Description
NBIA_SSH_CAI_PORT	SSH server login port Default value: 3322
NBIA_SSH_CAI_KEYSTORE_FILE ⁽¹⁾	SSH server authorization key store Default value: deploy/nbia_ssh_keystore.jks
NBIA_SSH_CAI_KEYSTORE_PASSWORD ⁽¹⁾	SSH server authorization key store password
NBIA_SSH_CAI_PRIVATEKEY_PASSWORD ⁽¹⁾	SSH server authorization private key store password
NBIA_SSH_CAI_KEYSTORE_ALIAS ⁽¹⁾	SSH server authorization key store alias Default value: nbia_ssh
NBIA_SSH_CAI_MACS	SSH server MACS Default value: hmacmd5,hmacmd596,hmacsha1,hmacsha196,macsha256,hmacsha512
NBIA_SSH_CAI_LOG_PATH	SSH CAI project log path Default value: /var/log/dve/mpe-business-nbi-ssh-cai.log
NBIA_SSH_CAI_LOGIN_GRACE_TIME	SSH CAI login grace time-out. If SSH server authorization time exceeds, an error is reported. Default value: 15s
NBIA_SSH_CAI_MAX_SESSION	SSH server node max session Default value:50
NBIA_SSH_CAI_CIPHER ⁽²⁾	SSH server ciphers applied Default value: aes128cbc,aes128ctr,arcfour128,blowfishcbc,tripledesCBC



Configuration Attribute Name	Description
NBIA_SSH_CAI_RESP_PROMPT	CAI interface response prompt Default value: blank space
NBIA_SSH_CAI_WELCOME_INFO	CAI interface welcome information Default value: *****welcome*****
NBIA_SSH_CAI_SESSION_IDLE_TIMEOUT	SSH CAI session idle time-out. If there is no data in the session, the session times out. Default value: 300000ms
NBIA_SSH_CAI_EXIT_CMD	SSH CAI interface exit command. Default value: exit
NBIA_SSH_CAI_CMD_PROMPT	SSH CAI command prompt Default value: blank space

(1) This parameter is recommended to be configured for security enhancement. For details, refer to Section 5.2.2 on page 19.

(2) The supported Cipher is restricted by JAVA JDK. To activate the unlimited cipher in NBIA SSH: 1. Download the JCE Unlimited Strength Jurisdiction Policy files from <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>; 2. Extract the jar files to the JDK home folder in EDA server: `${java.home}/jre/lib/security/`; 3. Restart all the nbia-module in EDA.

5.2.2 Adding/Changing Customized Properties for the Customized NBIA Submodule

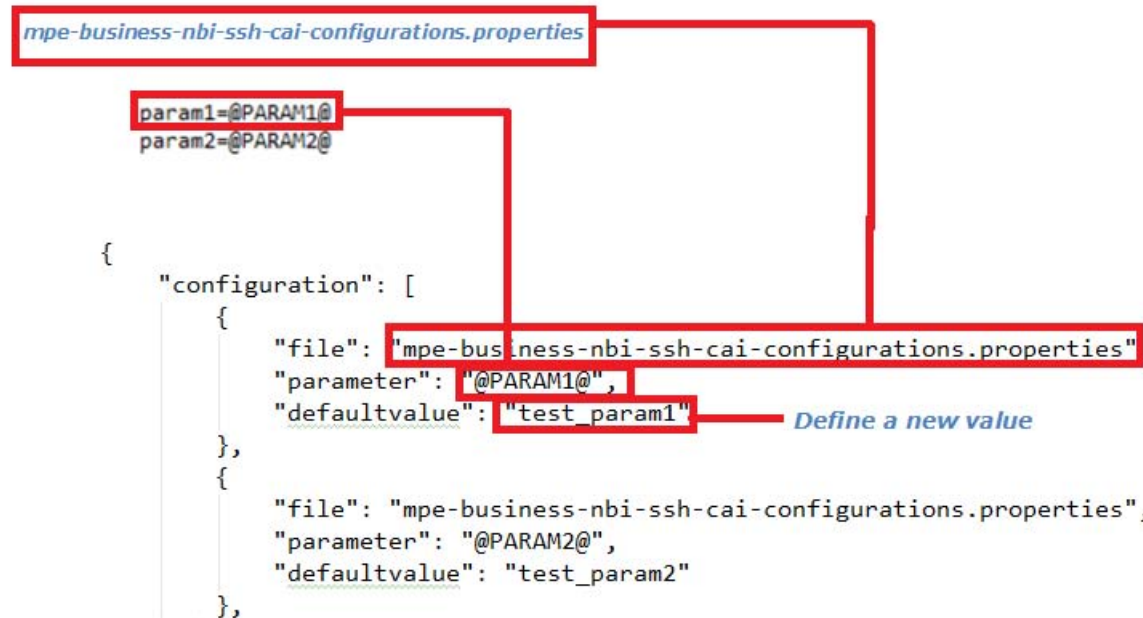
The properties are created or changed by editing the two configuration files in the following steps:

1. Create the property names in `mpe-business-nbi-ssh-cai-configurations.properties` as the following example.

```
param1=@PARAM1@
param2=@PARAM2@
```

2. Define the property values in `mpe-business-nbi-ssh-cai-module.conf`.

Edit the configuration block in `mpe-business-nbi-ssh-cai-module.conf` as follows:



"file" - the name of the configuration file that contains this property name

"parameter" - the name of the property

"defaultvalue" - the default value of the property

For existing parameters, changing the current default value during implementation also works.

For example, change "3322" to "3333" in the following example during implementation, and the default value of SSH port will be changed from "3322" to "3333" after packaging and deployment:

```
{
  "file": "mpe-business-nbi-ssh-cai-configurations.properties",
  "parameter": "@NBIA_SSH_CAI_PORT@",
  "defaultvalue": "3333"
},
```

5.2.3

Modifying Properties Values for the Customized NBIA Submodule

1. Run the following command from a system controller node to modify the value of a property after deployment:

```
# bootloader.py config set --parameter <parameter>
--value <value>
```

For example:



```
# bootloader.py config set --parameter @NBIA_SSH_CAI_PORT@ --value 3333
```

2. Run the following command from a system controller node to activate the configuration on PL nodes.

```
# bootloader node activate --host <hostname>
```

5.2.4 Restore the Default Value of the Property for the Customized NBIA Submodule

Note: The default values are defined in the `mpe-business-nbi-ssh-cai-module.conf` file.

1. Run the following command from a system controller node to restore the default value of a property:

```
# bootloader.py config remove --parameter <parameter>
```

For example:

```
# bootloader.py config remove --parameter @NBIA_SSH_CAI_PORT@
```

2. Run the following command from a system controller node to activate the configuration on PL nodes.

```
# bootloader node activate --host <hostname>
```

5.2.5 Key Store Configuration for the SSH CAI Submodule

Customer can generate a keystore for SSH CAI server to enhance security. For details, refer to the following documents:

- *System Administrators Guide for Native Deployment*, Reference [7]
- *System Administrators Guide for Virtual and Cloud Deployment*, Reference [6]

Note: Run the following commands as the `actadm` user. The following is an example using example-specific values, such as `-dname`. Choose your own parameters according to real environment.

1. Use the Java keystore tool to create a keystore.

```
$ /usr/java/latest/bin/keytool -genkey -keyalg RSA
-alias nbia_ssh -keystore /home/bootloader/ssl/nbia_ssh_keystore.jks -validity 3600 -keysize 2048
-dname 'CN=eg8-vip-trf.ete.ka.sw.ericsson.se, OU=IT, O=Ericsson, L=Unknown, ST=Stockholm, C=SE'
```

When prompted for keystore and key password, set the plain text values `<keystore password>` and `<key password>` respectively.



2. Run the following command to encrypt the plain text password:

Run the following command to get *<encrypted keystore password>*:

```
$ encrypt.sh '<keystore password>'
```

Run the following command to get *<encrypted key password>*:

```
$ encrypt.sh '<key password>'
```

3. Set bootloader keystore configuration for the submodule from a system controller node:

```
$ bootloader.py config set -p @NBIA_SSH_CAI_KEYSTORE_FILE@ -v /home/bootloader/ssl/<keystore file path>
```

```
$ bootloader.py config set -p @NBIA_SSH_CAI_KEYSTORE_PASSWORD@ -v <encrypted keystore code>
```

```
$ bootloader.py config set -p @NBIA_SSH_CAI_PRIVATEKEY_PASSWORD@ -v <encrypted key password>
```

```
$ bootloader.py config set -p @NBIA_SSH_CAI_KEYSTORE_ALIAS@ -v <keystore alias>
```

4. Perform an activation for the configured parameters to take effect on all modules of the system, one by one:

Note: Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter can only be used when no provisioning traffic is running.

Repeat the command for each SC node in the cluster.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

5.3 Configuring Logs

The SSH CAI submodule log file path is configured by parameter `@NBIA_SSH_CAI_LOG_PATH@`. To change the log level, edit the `mpe-business-nbi-ssh-cai-log4j.xml` file according to the following steps:

1. Open the configuration file `/usr/local/pgnngn/nbia-module-<version>/deploy/mpe-business-nbi-ssh-cai-log4j.xml` with a text editor.
2. Change the logger definition for Northbound Interface.

An example of changing log level to `WARN` is as follows:



```
<category name="com">
  <priority value="WARN"/>
</category>
```

3. Save the file. It takes some seconds before the change takes effect.
4. Repeat Step 1 to Step 3 for all Payload nodes in the system.

6 Verifying the Northbound Adapter

For the SSH CAI submodule, normal CAI requests can be applied for the verification of Northbound Interface, such as the HLR CAI request.

Follow the subsections to do verification.

6.1 Creating a User

The NBIA SSH interface applies public key authorization method. Therefore it needs one user that has a configured public key.

For information on how to add the public key, refer to section *Adding a User* in *User Guide for Resource Activation*, Reference [9].

The public key must be the one on the SSH client.

6.2 Login Through SSH Protocol

Log in from the SSH client to the Dynamic Activation server. For example, run the following command:

```
# ssh CAUser@10.170.13.28 -p 3322
```

```
The authenticity of host '[10.170.13.28]:3322'
([10.170.13.28]:3322) can't be established.
```

```
DSA key fingerprint is 9c:f1:a1:2e:ea:70:88:57:e5:6e:cf
:de:36:64:bc:5b.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '[10.170.13.28]:3322' (DSA)
to the list of known hosts.
```



If login to the SSH server is successful, the submodule deployment is successful. A welcome message and SSH prompt is displayed, for example:

```
*****welcome*****  
ssh-cai>
```

If login to the SSH server fails, the response message is shown in the following format:

```
[SSH message]:[error code]: [error message]
```

For example:

```
Received disconnect from 192.10.10.157: 4: Permission  
deny.
```

For details, refer to Section 6.4 on page 24.

6.3 Sending a CAI Request

To send a CAI request, for example, run the following command:

```
*****welcome*****  
  
ssh-cai> CREATE:HLRSUB:MSISDN,264000004010:IMSI,264000000  
04010;  
  
> <RESPONSE>
```

6.4 Error Code

This section contains the error codes.

Table 3 Error Code

Error Code	Error Message	Description
2	User session has timed out idling after <NBIA_SSH_CAI_SESSION_IDLE_TIMEOUT> ms.	The error may happen when the session idle times out. See Section 5.2.1 on page 18 for information.
4	Permission deny.	The error may happen during SSH login, for example, the user is invalid. See Section 6.2 on page 23 for more information about SSH login.



Error Code	Error Message	Description
12	Exceed the max value of user sessions.	The error may happen during SSH login when the user session exceeds the maximum value. See Section 6.1 on page 23 for information about how to set user sessions.
12	Exceed the max value of server sessions.	The error may happen during SSH login when the node session exceeds the maximum value. See Section 5.2.1 on page 18 for information about the <code><NBIA_SSH_CAI_MAX_SESSION></code> parameter.





Reference List

Ericsson Documents

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Customization - Architectural Overview*, 20/1553-CSH 109 628 Uen
- [3] *Northbound Interface Adapter Reference Manual*, 1/2134-CSH 109 628 Uen
- [4] *Customer Adaptation Development Guide for Resource Activation*, 5/1553-CSH 109 628 Uen
- [5] *CAI Interface Specification for HLR Components*, 24/155 19-CSH 109 628 Uen
- [6] *System Administrators Guide for Virtual and Cloud Deployment*, 3/1543-CSH 109 628 Uen
- [7] *System Administrators Guide for Native Deployment*, 1/1543-CSH 109 628 Uen
- [8] *Security and Privacy Management*, 0040-CSH 109 628 Uen
- [9] *User Guide for Resource Activation*, 1/1553-CSH 109 628 Uen
- [10] *ESA Fault Management*, 2/1543-FAM 901 455 Uen

Online Documents

- [11] *Understanding the SSH Encryption and Connection Process*,
<https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process>