

# Northbound Interface Adapter Reference Manual

Ericsson Dynamic Activation 1

---

## REFERENCE LIST

**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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	Typographic Conventions	1
1.4	Prerequisites	1
<b>2</b>	<b>Northbound Interface Adapter APIs</b>	<b>1</b>
2.1	Northbound	1
2.2	Service List	2
2.2.1	AlarmService	2
2.2.2	AuthenticationService	2
2.2.3	MBeanService	3
2.2.4	ProcLogService	4
2.2.5	ProvisionService	4
2.2.6	NBINotifyService	4
2.3	Data Model	5
2.3.1	ProcLogEntry	5
2.3.2	Request	7
2.3.3	Response	11
2.3.4	SessionData	11
2.3.5	Acknowledge	12
2.3.6	AsyncRequestMetaData	12
2.3.7	AsyncResponse	13
2.4	Exception	14
2.4.1	NBIException	14
2.4.2	NBIExternalException	15
2.4.3	NBIInternalException	15
	<b>Reference List</b>	<b>17</b>





# 1 Introduction

## 1.1 Purpose and Scope

This document is a reference for the development of northbound interface adapter in Ericsson™ Dynamic Activation (EDA).

## 1.2 Target Groups

The target group for this document is as follows:

- 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, OSGi, and Eclipse.

# 2 Northbound Interface Adapter APIs

There are more and more demands on customized northbound interfaces to meet the requirements from Customer Administration System (CAS) or order manager. To support customized northbound adapters, Dynamic Activation platform provides the following APIs so that customized northbound interfaces get to call Dynamic Activation platform services.

## 2.1 Northbound

Northbound interface is used for the customized northbound adapter to identify itself and get Dynamic Activation services.

**Table 1** Method Summary

Modifier and Type	Method and Description
void	<code>configure (Map&lt;String,String&gt; properties)</code>  This method is called by the Dynamic Activation platform when the configuration is updated for the northbound adaptor.
String	<code>getProtocol ()</code>  The identity of the northbound protocol, such as CAI, CAI3G, SOAP, etc.
void	<code>start (NBContext nbiContext)</code>  This method is called by the Dynamic Activation platform when the northbound adaptor is started.
void	<code>stop (NBContext nbiContext)</code>  This method is called by the Dynamic Activation platform when the northbound adaptor is stopped.

## 2.2 Service List

### 2.2.1 AlarmService

Northbound Interface Adapter (NBIA) Module Alarm is used for customized northbound adapter to send out events or alarms. For detailed information of Dynamic Activation system model, refer to *Event and Alarm Handling*, Reference [2].

The following table shows the `AlarmService` methods for NBIA.

**Table 2** Method Summary

Modifier and Type	Method and Description
<code>org.osgi.service.event.Event</code>	<code>ceaseAlarm(int errorCode, String errorMessage)</code>  Ceases an alarm
<code>org.osgi.service.event.Event</code>	<code>sendAlarm(int errorCode, String errorMessage)</code>  Sends an alarm
<code>org.osgi.service.event.Event</code>	<code>sendEvent(int errorCode, String errorMessage)</code>  Sends an event

### 2.2.2 AuthenticationService

`AuthenticationService` is used for the customized northbound adapter to authenticate the CAS login and logout requests towards Dynamic Activation user management.

**Table 3** Method Summary

Modifier and Type	Method and Description
boolean	<code>authenticate(String userName, String password)</code> Authenticates whether the user name with the password is valid. If the user is valid, value <code>true</code> is returned.
boolean	<code>authenticate(String userName, String password, String mode)</code> Authenticates whether the user name with the password is valid with mode type. If the user is valid, value <code>true</code> is returned. There are two traffic modes: <ul style="list-style-type: none"> <li>• Normal traffic mode: "" (Default value)</li> <li>• Test traffic mode: "Test"</li> </ul>
SessionContext	<code>getSession(String sessionId, String protocol)</code> Get the context of session bound to the session ID
String	<code>login(String userName, String password, String protocol)</code> Authenticates whether the user name with the password is valid. If the user is valid, a new session ID is returned and is added to session floating controller.
String	<code>login(String userName, String password, String protocol, String mode)</code> Authenticates whether the user name with the password is valid with mode type. If the user is valid, a new session ID is returned and is added to session floating controller.
void	<code>logout(String sessionId, String protocol)</code> Log out according to the session ID.

### 2.2.3

### MBeanService

MBeanService is used to provide JMX MBean service.

**Table 4** Method Summary

Modifier and Type	Method and Description
void	<code>registerMBean(StandardMBean mbean, String name)</code> Registers the <code>mbean</code> and uses the northbound interface name for the object name.
void	<code>unregisterMBean(String name)</code> Unregisters the <code>mbean</code> .



## 2.2.4 ProcLogService

`ProcLogService` is used for the customized northbound adapter to record processing logs.

*Table 5 Method Summary*

Modifier and Type	Method and Description
boolean	<code>isLevelThreeEnabled()</code> Indicates whether to record debug processing logs.
boolean	<code>isProclogEnabled()</code> Indicates whether to record processing logs.
void	<code>recordLog(ProcLogEntry procLogEntry, String component)</code> Saves a log item in the processing log file.

## 2.2.5 ProvisionService

`ProvisionService` is used for the customized northbound adaptor to execute and schedule the request.

*Table 6 Method Summary*

Modifier and Type	Method and Description
Response	<code>execute(Request request)</code> Executes the request synchronously.
Acknowledge	<code>schedule(Request request)</code> Schedules the request to execute it asynchronously.

Performance consideration:

- For the network protocol of customized northbound interface adapter such as SOAP, Jetty can handle multiple threads automatically, if internal HTTP server Jetty is used.
- For the network protocol of customized northbound interface adapter such as Telnet, multiple threads receive different TCP requests. Each TCP connection has a separate server thread to handle requests.

## 2.2.6 NBINotifyService

`NBINotifyService` defines the interface for asynchronous notification. The northbound asynchronous interface adapter needs to implement this interface `NBINotifyService`, and the platform calls this interface to send out the response after executing the request.





Table 7 Method Summary

Modifier and Type	Method and Description
Acknowledge	notify(AsyncResponse response) Handles the asynchronous response.

## 2.3 Data Model

### 2.3.1 ProcLogEntry

This is a holder class that contains the information about logs.

Table 8 Method Summary

Modifier and Type	Method and Description
void	cancel(int responseCode) Indicates this entry as a cancelled one; records stop time and response code.
void	cancel(String responseCode) Indicates this entry as a cancelled one; records stop time and response code.
static ProcLogEntry	createRootProcLogEntry(String protocol) Creates a root ProcLogEntry without start time.
static ProcLogEntry	createRootProcLogEntry(String protocol, boolean start) Creates a root ProcLogEntry with start time.
static ProcLogEntry	createNorthBoundProcLogEntry(String protocol) Create a northbound ProcLogEntry without start time.
static ProcLogEntry	createNorthBoundProcLogEntry(String protocol, boolean start) Create a northbound ProcLogEntry with start time.
static ProcLogEntry	createNorthBoundProcLogEntry(String protocol, String procLogId) Creates a northbound ProcLogEntry instance with provided procLogId (child ProcLogId in most cases if this API is used)
static ProcLogEntry	createNorthBoundProcLogEntry(String protocol, boolean start, String procLogId) Creates a northbound ProcLogEntry instance with start time and provided procLogId (child ProcLogId in most cases if this API is used).
void	fail() Indicates this entry as a failed one; records stop time.



Modifier and Type	Method and Description
void	<code>fail(int responseCode)</code> Indicates this entry as a failed one; records stop time and response code.
void	<code>fail(String responseCode)</code> Indicates this entry as a failed one; records stop time and response code.
ProcLogId	<code>getLogId()</code> Gets the object <code>ProcLogId</code> . It includes <code>rootId</code> and <code>subId</code> of the <code>proclog</code> .
String	<code>getInstance()</code> Gets the instance of this <code>ProcLogEntry</code> .
String	<code>getLogType()</code> Gets the <code>logType</code> of this <code>ProcLogEntry</code> .
String	<code>getOperation()</code> Gets the operation of this <code>ProcLogEntry</code> .
String	<code>getProcLogId()</code> Gets the <code>procLogId</code> of this <code>ProcLogEntry</code> .
String	<code>getProtocol()</code> Gets the protocol of this <code>ProcLogEntry</code> .
String	<code>getRequest()</code> Gets the request of this <code>ProcLogEntry</code> .
String	<code>getResponse()</code> Gets the response of this <code>ProcLogEntry</code> .
String	<code>getResponseCode()</code> Gets the <code>responseCode</code> of this <code>ProcLogEntry</code> .
long	<code>getStartTime()</code> Gets the <code>startTime</code> of this <code>ProcLogEntry</code> .
String	<code>getStatus()</code> Gets the status of this <code>ProcLogEntry</code> .
long	<code>getStopTime()</code> Gets the <code>stopTime</code> of this <code>ProcLogEntry</code> .
String	<code>getTarget()</code> Gets the target of this <code>ProcLogEntry</code> .
String	<code>getTransactionId()</code> Gets the <code>transactionId</code> of this <code>ProcLogEntry</code> .
String	<code>getUser()</code> Gets the user of this <code>ProcLogEntry</code> .
void	<code>setInstance(String instance)</code> Sets the instance of this <code>ProcLogEntry</code> .
void	<code>setLogType(String logType)</code> Sets the <code>logType</code> of this <code>ProcLogEntry</code> .



Modifier and Type	Method and Description
void	setOperation(String operation) Sets the operation of this ProcLogEntry.
void	setProtocol(String protocol) Sets the protocol of this ProcLogEntry.
void	setRequest(String request) Sets the request of this ProcLogEntry.
void	setResponse(String response) Sets the response of this ProcLogEntry.
void	setResponseCode(String responseCode). Sets the responseCode of this ProcLogEntry.
void	setStartTime(long startTime) Sets the startTime of this ProcLogEntry.
void	setStatus(String status) Sets the status of this ProcLogEntry.
void	setStopTime(long stopTime) Sets the stopTime of this ProcLogEntry.
void	setTarget(String target) Sets the target of this ProcLogEntry.
void	setTransactionId(String transactionId) Sets the transactionId of this ProcLogEntry.
void	setUser(String user) Sets the user of this ProcLogEntry.
void	start() Sets start time of the procLogEntry to current time.
void	succeed() Indicates this entry as a successful one; records stop time and assigns a default response code 0.
String	toString() Represents this log entry.

### 2.3.2 Request

`Request` is the general data model representing coming requests, and it defines several mandatory and optional attributes that are used by Dynamic Activation services. And also it provides a common format to carry the payload of the request.

**Table 9** *Attributes Used in Request*

Attribute	Type	Description
operation	String	operation usually means CRUD operations, but it can be any operation name.  For example: Create, Delete, Get, etc.
moName	String	moName is the logical name of a target MO. In CAI3G protocol, it is as same as MO type.  For example: HLRSUB, Subscription@http://schemas.ericsson.com/pg/auc/13.5/
moIds	Map	moIds means the MO identification, which needs to be unique identified target in BL.  For example: <code>{{imis, 494600000001}, {msisdn, 494600000001}}</code>
moAttributes	XML Document	Any XML format content that contains all required information for BL.  moAttributes is used to transfer data from northbound interface to proxy, and it is transparent to the current Dynamic Activation service, because it contains BL related information.
sessionData	SessionData	A set of session related data, such as user name, processing log ID, session ID  SessionData carries the session specific data, which is used by the Dynamic Activation framework.
rawRequest	String	Original requests sent from northbound interface  rawRequest stores the original requests from upstream, and it is used in the request search functions.
metadata	RequestMetadata	metadata is used to describe the attribute of the request. These attributes independent from the request protocol, payload type and session. These attributes are defined by framework for all requests and are independent from the request protocol, payload type and session.
testRequest	Boolean	testRequest indicates whether the request is in test mode.

The procedure of transporting provisioning data by Request is as follows:



1. Northbound interface receives the northbound requests and transforms the provisioning data to internal data model. `moAttributes` is used to save the internal data.
2. If you use your own interface and provisioning logic, save the northbound request data to `moAttributes`, and `moAttributes` is routed to the provisioning logic by Dynamic Activation.

The following example shows the CAI3G internal data model for the Create Request.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<MOAttributes>
  <CreateEPSMultiSC>
    <CreateEPSMultiSC imsi="494500100000" xmlns="http://schemas.ericsson.com/ma/HSS/">
      <imsi>494500100000</imsi>
      <epsProfileId>Profile900C</epsProfileId>
      <epsRoamingAllowed>true</epsRoamingAllowed>
    </CreateEPSMultiSC>
  </MOAttributes>
</Extension>
<NAMESPACE>
  default-ns-c9e9111b-fa9e-454a-b691-4578c8e14c27=http://schemas.ericsson.com/ma/HSS/ext/
</NAMESPACE>
</MOAttributes>
```

### Example 1 CAI3G Internal Data Model for the Create Request

The following example shows the CAI internal data model for the Create Request.

```
Command:
CREATE:HLRSUB:MSISDN,12340101:IMSI,56780101:PROFILE,0:BS31,1:BS32,1:BS33,1:BS34,
1:BS3G,1:BS3F,1:TS11,1:CFB,1,1,191019120003,OFA-1,1-A0AFBE01C001A:CFNRC,1,1,
191019120004,1-A0AFBE01C001A:CFNRY,1,1,191019120000,OFA-0:CFU,1,1,191019120000:DCF,
1,1,191019120000,20:SPN,1,1,191019120000:BAIC,1,1:BAOC,1,1:BICRO,1,1:BOIC,1,
1:BOIEXH,1,1:CAW,1,1;

<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<MOAttributes>
  <CreateHLRSUB xmlns="http://schemas.ericsson.com/ma/cai/1.0/">
    <MSISDN>12340101</MSISDN>
    <IMSI>56780101</IMSI>
    <PROFILE>0</PROFILE>
    <BS31>1</BS31>
    <BS32>1</BS32>
    <BS33>1</BS33>
    <BS34>1</BS34>
    <BS3G>1</BS3G>
    <BS3F>1</BS3F>
    <TS11>1</TS11>
    <CFB>
      <value id="1">1</value>
      <value id="2">1</value>
      <value id="3">191019120003</value>
      <value id="4">OFA-1</value>
      <value id="5">1-A0AFBE01C001A</value>
    </CFB>
    <CFNRC>
      <value id="1">1</value>
      <value id="2">1</value>
      <value id="3">191019120004</value>
      <value id="4">1-A0AFBE01C001A</value>
    </CFNRC>
    <CFNRY>
      <value id="1">1</value>
      <value id="2">1</value>
      <value id="3">191019120000</value>
    </CFNRY>
```



```
<value id="4">OFA-0</value>
</CFNRY>
<CFU>
  <value id="1">1</value>
  <value id="2">1</value>
  <value id="3">191019120000</value>
</CFU>
<DCF>
  <value id="1">1</value>
  <value id="2">1</value>
  <value id="3">191019120000</value>
  <value id="4">20</value>
</DCF>
<SPN>
  <value id="1">1</value>
  <value id="2">1</value>
  <value id="3">191019120000</value>
</SPN>
<BAIC>
  <value id="1">1</value>
  <value id="2">1</value>
</BAIC>
<BAOC>
  <value id="1">1</value>
  <value id="2">1</value>
</BAOC>
<BICRO>
  <value id="1">1</value>
  <value id="2">1</value>
</BICRO>
<BOIC>
  <value id="1">1</value>
  <value id="2">1</value>
</BOIC>
<BOIEXH>
  <value id="1">1</value>
  <value id="2">1</value>
</BOIEXH>
<CAW>
  <value id="1">1</value>
  <value id="2">1</value>
</CAW>
</CreateHLRSUB>
</MOAttributes>
</MOAttributes>
```

## Example 2 CAI Internal Data Model for the Create Request

Table 10 Method Summary

Modifier and Type	Method and Description
RequestMetadata	getMetadata() Gets the requestMetadata of the Request.
Document	getMOAttributes() Gets the moAttributes of the Request.
Map<String,String>	getMoId() Gets the moIds of the Request.
String	getMoName() Gets the moName of the Request.
String	getOperation() Gets the operation of the Request.
String	getRawRequest() Gets the rawRequest of the Request.



Modifier and Type	Method and Description
SessionData	getSessionData() Gets the sessionData of the Request.
boolean	isTestRequest() Check if the request is in test mode.
void	setMetadata(RequestMetadata metadata) Sets the requestMetadata of the Request.
void	setMOAttributes(Document moAttributes) Sets the moAttributes of the Request.
void	setMoId(Map<String,String> moIds) Sets the moIds of the Request.
void	setMoName(String moName) Sets the moName of the Request.
void	setOperation(String operation) Sets the operation of the Request.
void	setRawRequest(String rawRequest) Sets the rawRequest of the Request.
void	setSessionData(SessionData sessionData) Sets the sessionData of the Request.
void	setTestRequest(boolean testRequest) Sets the testRequest of the Request.

### 2.3.3

## Response

Table 11 Method Summary

Modifier and Type	Method and Description
Document	getMoAttributes() Gets the moAttributes of the Response.
void	setMoAttributes(Document moAttributes) Sets the moAttributes of the Response.

### 2.3.4

## SessionData

Table 12 Method Summary

Modifier and Type	Method and Description
String	getProclogId() Gets the proclogId of the SessionData.



Modifier and Type	Method and Description
Properties	<code>getProperties()</code> Gets the <code>SessionData</code> properties.
Object	<code>getProperty(String name)</code> Gets the property from the <code>SessionData</code> properties.
String	<code>getSessionId()</code> Gets the <code>sessionId</code> of the <code>SessionData</code> .
String	<code>getUsername()</code> Gets the username of the <code>SessionData</code> .
void	<code>setProclogId(String proclogId)</code> Sets the <code>proclogId</code> of the <code>SessionData</code> .
void	<code>setProperty(String name, Object value)</code> Sets the property of the <code>SessionData</code> .
void	<code>setSessionId(String sessionId)</code> Sets the <code>sessionId</code> of the <code>SessionData</code> .
void	<code>setUsername(String username)</code> Sets the username of the <code>SessionData</code> .

### 2.3.5

## Acknowledge

Table 13 Method Summary

Modifier and Type	Method and Description
BigInteger	<code>getCode()</code> Gets the code of <code>Acknowledge</code> .
String	<code>getMessage()</code> Gets the message of <code>Acknowledge</code> .
void	<code>setCode(BigInteger ackCode)</code> Sets the code of <code>Acknowledge</code> .
void	<code>setMessage(String ackMessage)</code> Sets the message of <code>Acknowledge</code> .

### 2.3.6

## AsyncRequestMetaData

Table 14 Method Summary

Modifier and Type	Method and Description
String	<code>getContext()</code> Gets the context of <code>AsyncRequestMetaData</code> .
String	<code>getFaultTo()</code> Gets the <code>faultTo</code> of <code>AsyncRequestMetaData</code> .





Modifier and Type	Method and Description
Date	getFireTime() Gets the fireTime of AsyncRequestMetaData.
String	getGroupId() Gets the groupId of AsyncRequestMetaData.
String	getMessageId() Gets the messageId of AsyncRequestMetaData.
String	getNotifyProtocol() Gets the notifyProtocol of AsyncRequestMetaData.
String	getPriority() Gets the priority of AsyncRequestMetaData.
String	getReplyTo() Gets the replyTo of AsyncRequestMetaData.
AsyncRequestMetaData	setContext(String context) Sets the context of AsyncRequestMetaData.
AsyncRequestMetaData	setFaultTo(String faultTo) Sets the faultTo of AsyncRequestMetaData.
AsyncRequestMetaData	setFireTime(Date fireTime) Sets the fireTime of AsyncRequestMetaData.
AsyncRequestMetaData	setGroupId(String groupId) Sets the groupId of AsyncRequestMetaData.
AsyncRequestMetaData	setMessageId(String messageId) Sets the messageId of AsyncRequestMetaData.
AsyncRequestMetaData	setNotifyProtocol(String notifyProtocol) Sets the notifyProtocol of AsyncRequestMetaData.
AsyncRequestMetaData	setPriority(String priority) Sets the priority of AsyncRequestMetaData.
AsyncRequestMetaData	setReplyTo(String replyTo) Sets the replyTo of AsyncRequestMetaData.

### 2.3.7 AsyncResponse

This class carries the response of the asynchronous command. It carries the original `com.ericsson.mpe.services.common.Request` and a normal `com.ericsson.mpe.services.common.Response` if the request runs successfully, or a `com.ericsson.mpe.common.error.NBIException` if the command fail.

*Table 15 Constructor Summary*

Constructor	Description
<code>AsyncResponse(Request request)</code>	Constructs a non-instated asynchronous response.
<code>AsyncResponse(Request request, NBIException exception)</code>	Constructs a failed asynchronous response.
<code>AsyncResponse(Request request, Response response)</code>	Constructs a successful asynchronous response.

*Table 16 Method Summary*

Modifier and Type	Method and Description
<code>void</code>	<code>fails(NBIException exception)</code> Set the failed exception of <code>AsyncResponse</code> .
<code>NBIException</code>	<code>getException()</code> Gets the exception of <code>AsyncResponse</code> .
<code>Request</code>	<code>getRequest()</code> Gets the request of <code>AsyncResponse</code> .
<code>Response</code>	<code>getResponse()</code> Gets the response of <code>AsyncResponse</code> .
<code>boolean</code>	<code>isFailed()</code> Check if the <code>AsyncResponse</code> failed.
<code>void</code>	<code>succeeds(Response response)</code> Set the successful response of <code>AsyncResponse</code> .
<code>ProcLogEntry</code>	<code>getProcLogEntry(String protocol)</code> Generates the <code>ProcLogEntry</code> of the <code>AsyncResponse</code> .

## 2.4 Exception

### 2.4.1 NBIException

This exception is used by northbound interface.

*Table 17 Constructor Summary*

Constructor	Description
<code>NBIException(long code, String description)</code>	Constructs the <code>NBIException</code> .
<code>NBIException(long code, String description, String addInfo)</code>	Constructs the <code>NBIException</code> .
<code>NBIException(long code, String description, String addInfo, Throwable throwable)</code>	Constructs the <code>NBIException</code> .
<code>NBIException(long code, String description, Throwable throwable)</code>	Constructs the <code>NBIException</code> .

**Table 18 Method Summary**

Modifier and Type	Method and Description
String	getAddInfo() Gets the addInfo of NBIException.
String	getDescription() Gets the description of NBIException.
long	getErrorCode() Gets the errorCode of NBIException.

## 2.4.2 NBIExternalException

This exception stands for external errors, such as the errors from `ElementManager` or `NetworkElement`, or Business Logic errors.

**Table 19 Constructor Summary**

Constructor	Description
<code>NBIExternalException(long code, String description)</code>	Constructs the NBIExternalException.
<code>NBIExternalException(long code, String description, String addInfo)</code>	Constructs the NBIExternalException.
<code>NBIExternalException(long code, String description, String addInfo, Throwable throwable)</code>	Constructs the NBIExternalException.
<code>NBIExternalException(long code, String description, Throwable throwable)</code>	Constructs the NBIExternalException.

## 2.4.3 NBIInternalException

This exception stands for internal component errors or framework problems.

**Table 20 Constructor Summary**

Constructor	Description
<code>NBIInternalException(long code, String description)</code>	Constructs the NBIInternalException.
<code>NBIInternalException(long code, String description, NBIInternalException.ErrorType type)</code>	Constructs the NBIInternalException.
<code>NBIInternalException(long code, String description, String addInfo, NBIInternalException.ErrorType type)</code>	Constructs the NBIInternalException.
<code>NBIInternalException(long code, String description, String addInfo, Throwable throwable, NBIInternalException.ErrorType type)</code>	Constructs the NBIInternalException.



Constructor	Description
<code>NBIInternalException(long code, String description, Throwable e)</code>	Constructs the <code>NBIInternalException</code> .
<code>NBIInternalException(long code, String description, Throwable throwable, NBIInternalException.ErrorType type)</code>	Constructs the <code>NBIInternalException</code> .

*Table 21 Method Summary*

Modifier and Type	Method and Description
<code>NBIInternalException.ErrorType</code>	<code>getType()</code> Gets the <code>ErrorType</code> .
<code>void</code>	<code>setType(NBIInternalException.ErrorType type)</code> Sets the <code>ErrorType</code> .



## Reference List

### Library Documents

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Event and Alarm Handling*, 3/1553-CSH 109 628 Uen