

DUP Migration API 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

1	Introduction	1
1.1	Purpose and Scope	2
1.2	Target Group	2
1.3	Typographic Conventions	2
1.4	Prerequisites	2
2	Data Type	2
3	Utilities Functions	3
3.1	ServiceContext	3
3.2	ResourceContext	3
3.3	RoutingUtils.registerNeTypes	4
3.4	TriggerMoRegister.addTriggerMoMapping	4
3.5	Cai3gMoMapping.unmarshallToMoBean	4
3.6	JAXBUtils.bean2Xml	5
4	DU Class	5
4.1	isDefined	5
4.2	IsOptional	6
4.3	SizeOf	7
4.4	Clear	8
5	DUString Class	9
5.1	checkString (Deprecated)	9
5.2	checkStr	11
5.3	convert2LowerCases	12
5.4	convert2UpperCases	12
5.5	currentTime	13
5.6	doubleQuote	14
5.7	findByTag	15
5.8	matchEre, getEreSubPattern, and undefineEre	16
5.9	getLine and removeFirstLine	17
5.10	int2str	18
5.11	length	19
5.12	matchRe	20



5.13	newline	20
5.14	readUntilFirstOfSeparators	21
5.15	readUntilRealToken	23
5.16	readUntilToken	24
5.17	replaceChar	25
5.18	replaceString	26
5.19	str2int	27
5.20	strip	28
5.21	subString	29
5.22	subStringByPos	31
6	NamedValue Class	32
6.1	print	32
6.2	newNamedValue	33
6.3	copy	34
6.4	delete	35
6.5	addValue	36
6.6	getValue	37
6.7	setValue	38
6.8	removeValue	39
6.9	getName	41
6.10	setName	42
6.11	addAttribute	44
6.12	getAttribute	45
6.13	setAttribute	46
6.14	attributes	47
6.15	getAttributeAt	48
6.16	addDu	50
6.17	get	51
6.18	getWithAttributes	53
6.19	entries	54
6.20	getAt	55
6.21	convert2ASN1	56
6.22	convert2Nv	58
6.23	buildFromString	59
6.24	convert2String	60
6.25	appendNamedValue	62



6.26	equal	63
7	ServiceLayer Class	64
7.1	process	64
7.2	lookup	66
8	ResourceLayer Class	67
8.1	doNeOperation	67
8.2	doNeRetrieveOperation	68
9	ResourceLayerGenerator Class	70
9.1	getFirstNamedValue	70
9.2	getFirstSubNamedValue	72
9.3	getNextNamedValue	73
9.4	getNextSubNamedValue	74
9.5	getValueWithName	75
10	ResourceLayerParser Class	76
10.1	setValue	76
10.2	setWholeNv	77
11	ConfigData Class	78
11.1	get	78
12	ConfigDataResourceLayer Class	80
12.1	get	80
13	Error Class	81
13.1	setCaiError	81
13.2	getCaiError	82
13.3	externalError	83
14	FDSStandardException	85
14.1	setErrorCode	85
14.2	getErrorCode	86
14.3	setReason	86
14.4	getReason	87
14.5	setHint	87
14.6	getHint	88
14.7	toJDVException	88
14.8	toFDSEException	89



15	Misc Class	90
15.1	exIsDefined	90
16	Build Class	91
16.1	convertToCai3g12Request	91
16.2	convertToCai3gRequest	93
	Reference List	97



1

Introduction

Ericsson™ Dynamic Activation (EDA) Data Unit Processing (DUP) is used for Classic Multi Activation BL development. This document provides API description for the JDVs converted from legacy DUP code. For details on CA in DUP from the Classic Multi Activation, refer to Multi Activation IDE Reference Manual, Reference [2].

The classes containing the JDV APIs in this document are described in the following table:

Table 1 Classes for JDV APIs

Class Name	Description
DU	DU is a data unit used to process the Java Bean related objects.
DUString	DUString is equivalent to String APIs in DUP that provide capability to manipulate String object in Java. Most of those APIs are based on the method in String Class.
NamedValue	NamedValue is equivalent to the NamedValue APIs in DUP that provide capability to manipulate NamedValue object. NamedValue is a data structure similar to XML DOM, which is commonly used in a DUP based project.
ServiceLayer	ServiceLayer is equivalent to Engine APIs in DUP that provide capability to retrieve routing or dispatch requests to another JDV.
ResourceLayer	ResourceLayer is equivalent to MML APIs in DUP that provide capability to dispatch requests from Service Layer JDV to Resource Layer JDV.
ResourceLayerGenerator	ResourceLayerGenerator is equivalent to MmlGenerator APIs in DUP that provide capability retrieve input XML stored in ResourceLayer Context. To align with DUP project, resource layer JDV stores the input and output in Resource Context and uses ResourceLayerGenerator and ResourceLayerParser to retrieve and put data in processing code.
ResourceLayerParser	ResourceLayerParser is equivalent to MmlParser APIs in DUP that provide capability to manipulate output XML stored in ResourceLayer Context. It works together with ResourceLayerGenerator to store and retrieve data in processing code.
ConfigData	ConfigData provides capacity to retrieve JDV configuration from GUI for Business Logic without recompilation.
Error	Error provides wrapper to set error code to FDSStandardException.
FDSStandardException	FDSStandardException is used for exceptions in the converted JDV project.



1.1 Purpose and Scope

This document is a reference for the APIs converted from DUP in Dynamic Activation. Not all DUP APIs have corresponding JDV APIs. Only the APIs in this document are supported.

1.2 Target Group

The target group for this document are as follows:

- Customer Adaptation (CA) developers
- Solution architects

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:

- Are familiar with Java.
- Have knowledge of DUP.
- Have knowledge of JDV CA development.

2 Data Type

The following table shows the data types used in DUP and Java.

Table 2 Data Types

DUP Type	Java Type	Change
VOID	void	No change
NULL	null	No change



DUP Type	Java Type	Change
BOOLEAN	Boolean	In DUP, the value of type BOOLEAN can be converted to the value of type INTEGER, with FALSE becoming 0 and TRUE becoming 1. In Java, this is not applicable.
ANY	Object	In DUP, ANY can be assigned with a variable of any data types. In Java, Object can only be assigned with a variable of the data types that are the Object sub-class.
INTEGER	Integer	No change
ENUMERATED	enum	No change
SET	ArrayList	No change
SEQUENCE	ArrayList	No change
IA5String	String	No change

3 Utilities Functions

Utilities function stand for the APIs that do not come from DUP ones. The functions support to build a workable JDV after it is translated from a DUP project. These functions are described in the following sections.

3.1 ServiceContext

`ServiceContext` is based on Thread Local to provide context information for service layer. The Context stores runtime information including `JDVConfiguration`, `Config Data`, `connector service`, and `neTypeMapping`.

For example:

```
ServiceContext.setDupRuntime(runtime);
Runtime runtime = ServiceContext.getDupRuntime();
```

Example 1 ServiceContext

3.2 ResourceContext

`ResourceContext` is based on Thread Local to provide context information for resource layer. The context stores runtime information including input request, output request, and connector service.



```
ResourceContext.getRuntime().setRequest(request);  
ResourceContext.getRuntime().setConnectorService(connectorService);  
Document respXml = ResourceContext.getRuntime().getOutputPayload();
```

Example 2 ResourceContext

3.3 RoutingUtils.registerNeTypes

`RoutingUtils.registerNeTypes` is used to register the mapping relation between `neType` and internal MO type. Those information is used when trying to find routing by a given internal MO.

```
Map<Class, String> neTypeMapping = new HashMap<>();  
neTypeMapping.put(Subscription.class, "NTYPE");  
neTypeMapping.put(AnotherSubscription.class, "AnotherNTYPE");  
RoutingUtils.registerNeTypes(neTypeMapping);
```

Example 3 RoutingUtils.registerNeTypes

3.4 TriggerMoRegister.addTriggerMoMapping

`TriggerMoRegister.addTriggerMoMapping` is used to register the mapping relation between trigger and internal MO operation. Those information is used when trying to unmarshall an input CAI3G request to an internal MO.

```
TriggerMoRegister.addTriggerMoMapping(Subscription.class, "Set", new QName(ADMPL, "SetSubscription"));  
TriggerMoRegister.addTriggerMoMapping(Subscription.class, "Get", new QName(ADMPL, "GetSubscription"));
```

Example 4 TriggerMoRegister.addTriggerMoMapping

3.5 Cai3gMoMapping.unmarshallToMoBean

`Cai3gMoMapping.unmarshallToMoBean` is used to unmarshall an input request payload to an internal MO based on standard JAXB according to the mapping relation between trigger and internal MO operation.

For example:

```
A request payload is:  
<records flags="adb">  
  <operation>0</operation>  
  <!--Optional:-->  
  <flags>true</flags>  
  <service>test service</service>  
</records>  
  
Records theMo = Cai3gMoMapping.unmarshallToMoBean(request);  
  
Result:  
theMo.flags == "true"  
theMo.flags.service == "test service"
```

Example 5 Cai3gMoMapping.unmarshallToMoBean



3.6 JAXBUutils.bean2Xml

JAXBUutils.bean2Xml provides the reverse operation to Cai3gMoMapping.unmarshallToMoBean. It is used to marhshall internal MO to XML document according to the trigger-MO mapping.

```
TempSubscription sub = new TempSubscription();
sub.setA3V(23);
sub.setCfnum("abc");
sub.setT22(1);
sub.setNewmsisdn("1234567");

Document doc = JAXBUutils.bean2Xml(sub);

Result:
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<TempSubscription>
  <newmsisdn>1234567</newmsisdn>
  <a3v>23</a3v>
  <t22>1</t22>
  <cfnum>abc</cfnum>
</TempSubscription>
```

Example 6 JAXBUutils.bean2Xml

4 DU Class

4.1 isDefined

Prototype:

```
public static boolean isDefined(Object obj);
```

Description:

This function is used to check if a variable is defined.

Corresponding API in DUP:

```
CONST BOOLEAN isDefined(theValue CONST ANY);
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Object mo	CONST ANY mo	The MO to be checked whether it is defined or not.	In	No change

Returned Value:

If `Object` is a Simple Variable, then `TRUE` is returned if it is set a value.
If `Object` is a Constructed Variable, then `TRUE` is returned if one of the parameters of the `Object` variable is set a value.

```
boolean isDef;  
Integer var = null;  
isDef = DU.isDefined(var); //isDef is false  
var = 10;  
isDef = DU.isDefined(var); // isDef is true
```

Example 7 isDefined

4.2 IsOptional

Prototype:

```
public static boolean isOptional(Object baseObj, String  
fieldName)
```

Description:

This function is used to check if a variable is optional.

Corresponding API in DUP:

```
CONST BOOLEAN isOptional(theValue CONST ANY);
```

Function Change:

No other change in function behavior.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Object baseObj	CONST ANY theValue (corresponding to baseObj.fieldName)	The MO to be checked whether it is optional.	In	No change
String fieldName	CONST ANY theValue (corresponding to baseObj.fieldName)	The field name of this MO	In	New

Returned Value:

This function returns TRUE if an Object fieldName is set to optional.

```
public class Subscription
{
    @XmlAttribute(required = true)
    protected String msisdn;
    protected Integer mpty;

    public String getMsisdn() {
        return msisdn;
    }

    public void setMsisdn(String msisdn) {
        this.msisdn = msisdn;
    }

    public Integer getMpty() {
        return mpty;
    }

    public void setMpty(Integer mpty) {
        this.mpty = mpty;
    }
}

boolean isOpt;
Subscription sub = new Subscription();
isOpt = DU.isOptional(sub, "mpty"); //isOpt is true
isOpt = DU.isOptional(sub, "msisdn"); //isOpt is false
```

Example 8 IsOptional

4.3

SizeOf

Prototype:

```
public static int sizeof(Object obj)
```

Description:

This function returns the number of elements in Object.

**Corresponding API in DUP:**

```
CONST INTEGER sizeof(theValue CONST ANY);
```

Function Change:

In Java, this function only supports `List` and `MO`.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Object mo	CONST ANY mo	The MO of which the size to be calculated	In	No change

Returned Value:

If the `Object` variable is a `List`, the number of the `List` items is returned.

If the `Object` variable is an `MO`, the number of fields is returned.

```
public class Subscription
{
    @XmlAttribute(required = true)
    protected String msisdn;
    protected Integer mpty;

    public String getMsisdn() {
        return msisdn;
    }

    public void setMsisdn(String msisdn) {
        this.msisdn = msisdn;
    }

    public Integer getMpty() {
        return mpty;
    }

    public void setMpty(Integer mpty) {
        this.mpty = mpty;
    }
}

Subscription sub = new Subscription();
int size = DU.sizeOf(sub); // size is 2
```

Example 9 SizeOf

4.4

Clear

Prototype:

```
public static void clear(T Object obj) throws
FDSStandardException
```



Description:

The function clears the variable `obj` to make it undefined. If the variable has a default value defined, the variable gets that value.

Corresponding API in DUP:

```
VOID clear(theVariable ANY);
```

Function Change:

This function is applicable for MO only and is not applicable for simple data type.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
T Object <code>obj</code>	<code>theVariable ANY</code>	The MO to be cleared.	In, out	No change

Returned Value:

None

```
Subscription sub = new Subscription();
sub.setMsisdn("11112222"); //the msisdn in sub is "11112222"
sub.setMpty(123);         //the mpty in sub is 123
DU.clear(sub);             //the msisdn in sub is "" and mpty is null
```

Example 10 Clear

5 DUString Class

5.1 checkString (Deprecated)

Prototype:

```
public static int checkString(String checktype, String
str)
```

**Description:**

This API is deprecated and replaced by `checkStr()`, see Section 5.2 on page 10.

Corresponding API in DUP:

```
INTEGER String::checkString(CONST IA5String checktype,  
CONST IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String checktype	CONST IA5String checktype	The definition of the type	In	No change
String str	CONST IA5String string	The string to be checked	In	No change

Returned Value:

The result of the check: 1 means the string matches the `checktype`, otherwise it returns 0.

Note: There are four types of `checktype`:

- `isalpha`: checks whether all the characters in the string are letters or not.
- `isdigit`: checks whether all the characters in the string are digits or not.
- `isinteger`: checks whether the string is an integer or not.
- `isxdigit`: checks whether all the characters in the string are hexadecimal digit or not.

If the `checktype` is not the standard type (`isalpha`, `isdigit`, `isinteger`, `isxdigit`), 1 is always returned.

```
int checked = DUStrng.checkString("isalpha", "abCD"); //checked is 1  
checked = DUStrng.checkString("isdigit", "123"); //checked is 1  
checked = DUStrng.checkString("isxdigit", "0F1"); //checked is 1
```

Example 11 `checkString`



5.2 checkStr

Prototype:

```
public static boolean checkStr(String checktype, String str)
```

Description:

The function checks the type of a string.

Corresponding API in DUP:

```
INTEGER String::checkString(CONST IA5String checktype, CONST IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String checktype	CONST IA5String checktype	The definition of the type	In	No change
String str	CONST IA5String string	The string to be checked	In	No change

Returned Value:

The result of the check: `true` means the string matches the `checktype`, otherwise it returns `false`.

Note: There are four types of `checktype`:

- `isalpha`: checks whether all the characters in the string are letters or not.
- `isdigit`: checks whether all the characters in the string are digits or not.
- `isinteger`: checks whether the string is an integer or not.
- `isxdigit`: checks whether all the characters in the string are hexadecimal digit or not.

If the `checktype` is not the standard type (`isalpha`, `isdigit`, `isinteger`, `isxdigit`), `true` is always returned.



```
boolean checked = DUString.checkString("isalpha", "abCD"); //checked is true
checked = DUString.checkString("isdigit", "123"); //checked is true
checked = DUString.checkString("isxdigit", "0F1"); //checked is true
```

Example 12 *checkStr*

5.3 convert2LowerCases

Prototype:

```
public static String convert2LowerCases(String str)
```

Description:

The function converts string characters to lower cases.

Corresponding API in DUP:

```
IA5String String::convert2LowerCases(CONST IA5String
string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	CONST IA5String string	The string to be converted	In	No change

Returned Value:

The converted string.

```
String str = DUString.convert2LowerCases("AaBbCcDd"); //str is "aabbccdd"
```

Example 13 *convert2LowerCases*

5.4 convert2UpperCases

Prototype:

```
public static String convert2UpperCases(String str)
```

**Description:**

The function converts string characters to upper cases.

Corresponding API in DUP:

```
IA5String String::convert2UpperCases(CONST IA5String
string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	CONST IA5String string	The string to be converted	In	No change

Returned Value:

The converted string.

```
String str = DUString.convert2UpperCases("AaBbCcDd") //str is "AABBCCDD"
```

Example 14 convert2UpperCases

5.5 currentTime

Prototype:

```
public static String currentTime(String time)
```

Description:

The function generates the current time as a string.

Corresponding API in DUP:

```
IA5String String::currentTime(CONST IA5STRING template)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String time	CONST IA5STRING template	The format of the output	In	No change

Returned Value:

The current time formatted according to the template.

```
String time = DUString.currentTime("%Y-%m-%d %H:%M");  
//return the current time like "2015-11-3 14:11"
```

Example 15 *currentTime*

5.6 doubleQuote

Prototype:

```
public static String doubleQuote()
```

Description:

The function generates a double quote.

Corresponding API in DUP:

```
IA5String String::doubleQuote()
```

Function Change:

No other change in function behavior.

Parameter Table:

None

Returned Value:

A double quote.

```
String dQuote =DUString.doubleQuote(); //dQuote is "\""
```

Example 16 *doubleQuote*



5.7 findByTag

Prototype:

```
public static String findByTag(String tag, String str)
```

Description:

The function finds a sub-string in a string. The sub-string follows a certain tag and ends by the end of the string.

Corresponding API in DUP:

```
IA5String String::findByTag(CONST IA5String tag, CONST  
IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String tag	CONST IA5String tag	The tag that can be a character or a string.	In	No change
String str	CONST IA5String string	The input string that contains the tag.	In	No change

Returned Value:

The sub-string.

Note:

- If a character before the tag is a letter, the tag does not take effect.
- If a tag is not contained in a string, an empty string ("") is returned.
- If a tag is at the beginning of the string, it takes effect.
- If "\t" or "\n" is located after a tag in a string, the sub-string is between this tag and the character "\t" or "\n".
- If a string contains more than one tag, only the last tag takes effect.



```
String str = DUString.findByTag("/", "abcd"); //str is "abcd"  
str = DUString.findByTag("/", "a/");//str is ""  
str = DUString.findByTag("ca", "a23cab\tal");//str is "b"  
str = DUString.findByTag("ca", "a23cab\nal");//str is "b"
```

Example 17 *findByTag*

5.8 matchEre, getEreSubPattern, and undefineEre

Prototype:

```
public static int matchEre(Integer ereId, String pattern,  
String string)  
  
public static String getEreSubPattern(Integer ereId,  
Integer nbr)  
  
public static void undefineEre(Integer erId)
```

Description:

`matchEre` matches an Extended Regular Expression (ERE) in a string.

`getEreSubPattern` gets the matched sub-string from an array that is generated by `String::matchEre`.

`undefineEre` removes the array generated by `String::matchEre`.

Corresponding API in DUP:

```
INTEGER String::matchEre(CONST INTEGER EreId, CONST  
IA5String pattern, CONST IA5String string)
```

```
IA5String String::getEreSubPattern(CONST INTEGER EreId,  
CONST INTEGER Nbr)
```

```
VOID String::undefineEre(CONST INTEGER EreId)
```

Function Change:

`matchEre` in DUP is based on standard C API `regcomp` while in Java it is based on `java.util.regex.Matcher`. Although both methods support POSIX regular expression, there could be some small difference according to the reference: `regcomp` <http://pubs.opengroup.org/onlinepubs/009695399/functions/regcomp.html> and `Java Matcher` <http://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html>



For `getEreSubPattern`, in both DUP and Java, the returned value means whether the pattern is matched, but in Java, the returned value also means the array identifier.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Integer <code>ereId</code>	CONST INTEGER <code>EreId</code>	The identifier of the array that stores the matched sub-string.	In	No change
Integer <code>nbr</code>	CONST INTEGER <code>Nbr</code>	The index number of the matched string stored in the array	In	No change
String <code>pattern</code>	CONST IA5String <code>pattern</code>	The ERE pattern.	In	No change
String <code>string</code>	CONST IA5String <code>string</code>	The string to be checked.	In	No change

Returned Value:

-1 means the pattern is not matched.

Other values mean the pattern is matched and the array identifier.

```
String str = "SKEY=9999,DEFCH=R,SKEY=30,SKEY=888,UT=N;";
String pattern = "SKEY=([0-9]+|N)";
int id = DUString.matchEre(null, pattern, str);
String matchedString = DUString.getEreSubPattern(id,1); //matchedString is "9999"
DUString.undefineEre(id);
```

Example 18 matchEre, getEreSubPattern, and undefineEre

5.9 getLine and removeFirstLine

Prototype:

```
public static String getline(String str)
```

```
public static String removeFirstLine(String str)
```

Description:

The function gets the first line of a string.

Corresponding API in DUP:

```
IA5String String::getline(IA5String string)
```

**Function Change:**

The first line in the parameter `string` is removed when the `getLine()` method in DUP is called, but in Java the string without the first line is returned only when the method `removeFirstLine()` is called.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String <code>str</code>	IA5String <code>string</code>	The string from which to get the first line.	In	The method parameter in DUP is IN and OUT, while the parameter in Java is IN only.

Returned Value:

For public static String `getline(String str)`, the returned value is the first line of the string.

For public static String `removeFirstLine(String str)`, the returned value is the string from which the first line is stripped.

```
String str = "1st line\n2ed line";
String firstLine = DUStrng.getline(str); // firstLine is "1st line"
str = DUStrng.removeFirstLine(str); // str is "\n2ed line"
```

Example 19 *getline*

5.10 int2str

Prototype:

```
public static String int2str(Integer integer)
```

Description:

The function converts an integer to a string.

Corresponding API in DUP:

```
IA5String String::int2str(CONST INTEGER int)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Integer integer	CONST INTEGER int	The integer to be converted.	In	No change

Returned Value:

The string converted from the integer.

```
String str = DUString.int2str(16); //str is "16"
```

Example 20 int2str

5.11 length

Prototype:

```
public static Integer length(String str)
```

Description:

The function returns the length of a string.

Corresponding API in DUP:

```
INTEGER String::length(CONST IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	CONST IA5String string	The string to be checked.	In	No change

Returned Value:

The length of the string.

```
Integer len = DUString.length("aabbccdd") //len is 8
```

Example 21 length



5.12 matchRe

Prototype:

```
public static boolean matchRe(String pattern, String str)
```

Description:

The function matches a Regular Expression (RE) in a string and only returns a boolean result.

Corresponding API in DUP:

```
INTEGER String::matchRe(CONST IA5String pattern, CONST  
IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String pattern	CONST IA5String pattern	The RE pattern.	In	No change
String str	CONST IA5String string	The string to be checked.	In	No change

Returned Value:

If an RE is matched, `true` is returned. Otherwise, `false` is returned.

```
String str = "FAULT CODE 14 abc";  
boolean matched = DUStrng.matchRe("FAULT CODE +[0-9]+ +[a-z]+", str); //matched is true
```

Example 22 matchRe

5.13 newline

Prototype:

```
public static String newline()
```

**Description:**

The function generates a new line.

Corresponding API in DUP:

```
IA5String String::newline( )
```

Function Change:

No other change in function behavior.

Returned Value:

The string "\n".

```
String str = DUString.newline() // str is "\n"
```

Example 23 newline

5.14 readUntilFirstOfSeparators

Prototype:

```
public static String[] readUntilFirstOfSeparators(String
strIn, String[] sepSets, String strType, String separator)

public static String getFirstSeparator()

public static String getSubStr()
```

Description:

This function reads the separators from a string until the first of the supplied separators is found. Tokens that are separated by the first separator are returned.

Corresponding API in DUP:

```
IA5STRING String::readUntilFirstOfSeparators(IA5STRING
string, CONST ANY separators, CONST IA5STRING stringtype,
IA5STRING firstseparator)
```

Function Change:

In Java, the original DUP function is divided into three functions:



- `readUntilFirstOfSeparators()` returns a set of `String` that is used to get the value of the first matched separator. The first matched separator separates the substring into two parts.
- `getFirstSeparator()` is used to return the first matched separator.
- `getSubStr()` is used to return the second part of the substring.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
<code>String strIn</code>	IA5STRING string	The string to read token from and the returned token is stripped from the string.	In	The first method parameter in DUP is IN or OUT, while the parameter in Java is IN only. This parameter can be collected by the <code>getSubStr()</code> method.
<code>String[] sepSets</code>	None	A set of token separators, including normal separators and DQUOTE = "\" EOS =End Of String. EOS MUST be the last item in a list if it is used.	In	This parameter returns a set of <code>String</code> , including the first matched separator, the first part of the substring and the second part of the substring separated by the separator.
<code>String strType</code>	CONST IA5STRING stringtype	The type of string to search for "STRING" means the string that is not surrounded by double quotes. "FULL_STRING" means the string that is surrounded by double quotes.	In	No change
<code>String separator</code>	IA5STRING firstseparator	The first found separator (from the set of separators) in the string.	In	The first method parameter in DUP is IN or OUT, while the parameter in Java is IN only. This parameter can be collected by the <code>getFirstSeparator()</code> method.

Returned Value:

A set of `String` with three parameters. For details, see the function change.

`getFirstSeparator()`: the first found separator.



`getSubStr()`: the second part of the substring separated by the separator.

Note:

- If `strType` is “FULL_STRING”, the separator can be either in double quotes or without them. See Example 24 for detailed information.
- If the separator is “EOF”, the function skips it.

```
String str = "123:456,789;abc";
String [] sepSets = {"", ",", ":", ";"};
DUString.readUntilFirstOfSeparators(str, sepSets, "FULL_STRING", "");
String separator = DUString.getFirstSeparator();//separator is ":"
String sub1stStr = DUString.getToken();//sub1stStr is "123"
String sub2ndStr = DUString.getSubStr();//sub2ndStr is "456,789;abc"

str = "\"123\":456,789;abc";
DUString.readUntilFirstOfSeparators(str, sepSets, "FULL_STRING", "");
separator = DUString.getFirstSeparator();//separator is "\"123\""
sub1stStr = assertEquals(DUString.getToken();//sub1stStr is ""
sub2ndStr = assertEquals(DUString.getSubStr();//sub2ndStr is ":456,789;abc"
```

Example 24 readUntilFirstOfSeparators

5.15 readUntilRealToken

Prototype:

```
public static String readUntilRealToken(String strIn,
String token)

public static String readUntilRealTokenSecondPart (String
strIn, String token)
```

Description:

This function separates a string with a separator, and it returns the first part.

Note: If the separator is in a pair of double quotes, it does not take effect.

Corresponding API in DUP:

```
IA5String String::readUntilRealToken(IA5String string,
IA5String separator)
```

Function Change:

The `String` parameter of the `readUntilRealToken` method in DUP can be changed after the method is called, but in Java the referred parameter `strIn` can be changed only when the method `readUntilRealTokenSecondPart()` is called.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String strIn	IA5STRING string	The string to be separated.	In	The first method parameter in DUP can be IN or OUT, while the parameter in Java is IN only.
String token	IA5STRING separator	The separator	In	No change

Returned Value:

For public static String readUntilRealToken(String strIn, String token), the returned value is the sub-string before the separator.

For public static String readUntilRealTokenSecondPart(String strIn, String token), the returned value is the sub-string after the separator.

```
String strIn = "123/45";
String firstP = DUStrng.readUntilRealToken(strIn, "/"); // firstP is "123"
strIn = DUStrng.readUntilRealTokenSecondPart(strIn, "/"); // strIn is "45"

strIn = "\"12\\\"A\\\"34\\\"A\\\"45A6\"";
String firstP = DUStrng.readUntilRealToken(strIn, "A"); // firstP is "12\\\"A\\\"34\\\"A\\\"45"
strIn = DUStrng.readUntilRealTokenSecondPart(strIn, "A"); // strIn is "6"
```

Example 25 readUntilRealToken

5.16 readUntilToken

Prototype:

```
public static String readUntilToken(String strIn, String token)
```

```
public static String readUntilTokenSecondPart(String strIn, String token)
```

Description:

The tokens are read from a string, and the returned token is striped off from the string.

Corresponding API in DUP:

```
IA5String String::readUntilToken(IA5String string, CONST IA5String token)
```



Function Change:

The token before the separator in the parameter `string` is removed when the `readUntilToken()` method in DUP is called, but in Java the token after the separator is returned only when the method `readUntilTokenSecondPart()` is called.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String <code>strIn</code>	IA5STRING <code>string</code>	The string to be divided.	In	The first method parameter in DUP can be IN or OUT, while the parameter in Java is IN only.
String <code>token</code>	CONST IA5String <code>token</code>	The token treated as the separator.	In	No change

Returned Value:

For `public static String readUntilToken(String strIn, String token)`, the returned value is the sub-string before the separator.

For `public static String readUntilTokenSecondPart(String strIn, String token)`, the returned value is the sub-string after the separator.

```
String strIn = "aAa123";
String str = DUString.readUntilToken(strIn, "A"); //str is "a"
strIn = DUString.readUntilTokenSecondPart(strIn, "A"); // strIn is "a123"
```

Example 26 readUntilToken

5.17 replaceChar

Prototype:

```
public static String replaceChar(String str, char od, char nw)
```

Description:

This function replaces all the occurrence old chars (ASCII code) with the new chars in a string.

**Corresponding API in DUP:**

```
IA5STRING String::replaceChar(CONST IA5STRING string,  
CONST IA5STRING old, CONST IA5STRING new)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String <i>str</i>	CONST IA5STRIN G <i>string</i>	The string to read from.	In	No change
char <i>od</i>	CONST IA5STRIN G <i>old</i>	The ASCII code of the old char.	In	No change
char <i>nw</i>	CONST IA5STRIN G <i>new</i>	The ASCII code of the new char.	In	No change

Returned Value:

The replaced string.

Type: String

```
String str = DUString.replaceChar("abce1234ABab",'a' , 'C'); //str is "Cbce1234ABCb"
```

Example 27 replaceChar

5.18 replaceString

Prototype:

```
public static String replaceString(String str, String  
oldStr, String newStr)
```

Description:

This function replaces a sub-string in a string with the new one.

Corresponding API in DUP:

```
IA5String String::replaceString(IA5String string, CONST  
IA5String old, CONST IA5String new)
```


**Function Change:**

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	IA5String string	The string from which a sub-string is replaced.	In	No change
char oldStr	CONST IA5String old	The sub-string to be replaced.	In	No change
char newStr	CONST IA5String new	The new sub-string.	In	No change

Returned Value:

The string whose sub-string has been replaced.

```
String str = DUString.replaceString("abce1234ABab","ab" , "C5"); //str is "C5ce1234ABC5"
```

Example 28 replaceString

5.19

str2int**Prototype:**

```
public static int str2int(String str)
```

Description:

This function converts a string to an integer.

Corresponding API in DUP:

```
INTEGER String::str2int(CONST IA5String string)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	CONST IA5String string	The string to be converted.	In	No change

**Returned Value:**

The integer converted from the string.

```
Integer i = DUString.str2int("123");// i is 123
```

Example 29 *str2int*

5.20

strip

Prototype:

```
public static String strip(String str, String target,  
String method)
```

Description:

This function strips a certain character from the beginning and the end of a string.

Corresponding API in DUP:

```
IA5String String::strip(CONST IA5String string, CONST  
IA5String char, CONST IA5String method)
```

Function Change:

Java does not support \v.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String <i>str</i>	CONST IA5String <i>string</i>	The string from which a character is stripped.	In	No change
String <i>target</i>	CONST IA5String <i>char</i>	The character that is to be stripped.	In	No change
String <i>method</i>	CONST IA5String <i>method</i>	The stripping method	In	No change

Returned Value:

The string from which the character is stripped.

**Note:**

- The second argument "char" must be a single character, otherwise only the first character of the argument takes effect.
- The second argument "char" can be set to some tokens that stand for some special characters. The default value is " " (the blank space).

NL = \n

HT = \t

CR = \r

FF = \f

DQUOTE = \"

- There are three types of methods to strip the character:
 - LEADING: only strip the matched character at the beginning of the string.
 - TRAILING: only strip the matched character at the end of the string.
 - BOTH: strip the matched character at both the beginning and the end of the string.

The default method is BOTH.

```
String str = "AbceAdfghA";
String tmp = DUString.strip(str, "A", "BOTH"); //tmp is "bceAdfgh"
tmp = DUString.strip(str, "A", "LEADING"); //tmp is "bceAdfghA"
tmp = DUString.strip(str, "A", "TRAILING"); //tmp is "AbceAdfgh"
```

Example 30 strip

5.21 subString

Prototype:

```
public static String subString(String str, String head,
String tail)
```

Description:

This function gets a sub-string that starts from the head token and ends before the tail token of a string.

**Corresponding API in DUP:**

```
IA5String String::subString(IA5String string, CONST  
IA5String head, CONST IA5String tail)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	IA5String string	The string from which a sub-string is retrieved.	In	No change
String head	CONST IA5String head	The first token.	In	No change
String tail	CONST IA5String tail	The second token.	In	No change

Returned Value:

The sub-string that starts from the head token and ends before the tail token.

If the head token cannot be matched in the string, an empty string ("") is returned.

Type: String

Note:

- The head token can be a character or a string. If the head token is a string, only the first character is treated as the real token.
- The tail token can be a character or a string. If the tail token is a string, only the last character is treated as the real token.
- The sub-string contains the head token but not the tail token.
- If the head token cannot be matched in a string, "" is returned.
- If the tail token cannot be matched in a string but the head token can be matched, the sub-string starts from the head token and ends until the end of the string.

```
String str = "abacdefg";  
String result = DString.subString(str, "a", "f"); //result is "abacde"  
result = DString.subString(str, "ab", "ef"); //result is "abacde"  
result = DString.subString(str, "ef", "ab"); //result is ""
```

Example 31 *subString*



5.22 subStringByPos

Prototype:

```
public static String subStringByPos(String str, Integer bPos, Integer ePos)
```

Description:

This function gets the sub-string in a string between the beginning and the ending position.

Corresponding API in DUP:

```
IA5String String::subStringByPos(IA5String string, CONST  
INTEGER b_pos, CONST INTEGER e_pos)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String str	IA5String string	The string from which a sub-string is received.	In	No change
Integer bPos	CONST INTEGER b_pos	The beginning position of the sub-string.	In	No change
Integer ePos	CONST INTEGER e_pos	The ending position of the sub-string	In	No change

Returned Value:

The sub-string that starts at the beginning position (defined by `b_pos`) and ends at the ending position (defined by `e_pos`) of a string.

**Note:**

- If both values of the `b_pos` and `e_pos` are greater than the string length, the sub-string equals to "".
- If the value of `e_pos` equals to 0 or is greater than the string length, the sub-string starts from `b_pos` and ends by the end of the string.
- If the value of `e_pos` is smaller than the value of `b_pos`, the values of the two variables are exchanged.

```
String str = "abcdefg";  
String result = DUStrng.subStringByPos(str, 0, 3); //result is "abcd"  
result = DUStrng.subStringByPos(str, 0, 0); //result is "abcdefg"  
result = DUStrng.subStringByPos(str, 1, 15); //result is "bcdefg"  
result = DUStrng.subStringByPos(str, 6, 3); //result is "defg"
```

Example 32 *subStringByPos*

6 NamedValue Class

6.1 print

Prototype:

```
public static void print(int namedvaluePtr) throws  
FDSStandardException
```

Description:

This function prints out the `NamedValue` structure.

Corresponding API in DUP:

```
VOID NamedValue::print(CONST INTEGER namedvaluePtr)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change

Returned Value:

None

Exception:

FDSStandardException is thrown if the oldname cannot be found.

6.2 newNamedValue

Prototype:

```
public static int newNamedValue(String name, String value)
```

Description:

This function creates a new NamedValue structure.

Corresponding API in DUP:

```
INTEGER NamedValue::new(CONST IA5String name, CONST IA5String value)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5String name	The name of the tag.	In	No change
String value	CONST IA5String value	The value of the tag.	In	No change

**Returned Value:**

The pointer of the NamedValue structure.

Type: int

Note:

- The generated NamedValue structure must be released after it is used.
- The second parameter value can be a null string.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList></respList>
```

Example 33 newNamedValue

6.3 copy

Prototype:

```
public static int copy(int namedValuePtr)
```

Description:

This function copies a NamedValue structure to another one.

Corresponding API in DUP:

```
INTEGER NamedValue::copy(CONST INTEGER namedvaluePtr)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedValuePtr	CONST INTEGER namedvaluePtr	The pointer of the source NamedValue structure.	In	No change

**Returned Value:**

The pointer of the target NamedValue structure.

Type: int

Note: The copied NamedValue structure must be released after it is used.

```
int ptr = NamedValue.newNamedValue("respList", "");
int tPtr = NamedValue.copy(ptr);
NamedValue.print(tPtr);
NamedValue.delete(ptr);
NamedValue.delete(tPtr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList></respList>
```

Example 34 copy

6.4

delete

Prototype:

```
public static void delete(int namedValuePtr)
```

Description:

This function deletes a NamedValue structure.

Corresponding API in DUP:

```
VOID NamedValue::delete(CONST INTEGER namedvaluePtr)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedValuePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change

Returned Value:

None



6.5 addValue

Prototype:

```
public static void addValue(int ptr, String name, String value)
```

Description:

This function adds an entry into the `NamedValue` structure.

Corresponding API in DUP:

```
VOID NamedValue::addValue(CONST INTEGER namedvaluePtr,  
CONST IA5String name, CONST IA5String value)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int <code>ptr</code>	CONST INTEGER <code>namedvaluePtr</code>	The pointer of the <code>NamedValue</code> structure into which a new entry is added.	In	No change
String <code>name</code>	CONST IA5String <code>name</code>	The entry name (including the full path).	In	No change
String <code>value</code>	CONST IA5String <code>value</code>	The entry value.	In	No change

Returned Value:

None



```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "value");
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>value</III>
  </II>
</respList>
```

Example 35 addValue

6.6 getValue

Prototype:

```
public static String getValue(int ptr, String name) throws
FDSStandardException
```

Description:

This function retrieves a value according to the given `ptr` and `name`.

Corresponding API in DUP:

```
IA5String NamedValue::getValue(CONST INTEGER namedvalue
Ptr, CONST IA5String name)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
<code>int ptr</code>	CONST INTEGER <code>namedvaluePtr</code>	The pointer of the NamedValue structure.	In	No change
<code>String name</code>	CONST IA5String <code>name</code>	The entry name (including the full path).	In	No change

Returned Value:

The value of the entry.



Note: If the second parameter name is "", the value of the first level entry of the NamedValue structure is returned.

Exception:

FDSStandardException is thrown if the valuenam cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "");
NamedValue.addValue(ptr, "II.III.IIII", "value");
NamedValue.getValue(ptr, "II.III.IIII"); // return is "value"
NamedValue.getValue(ptr, "");           // return is ""
NamedValue.delete(ptr);
```

Example 36 *getValue*

6.7 setValue

Prototype:

```
public static void setValue(int ptr, String name, String
value) throws FDSStandardException
```

Description:

This function sets the value of an entry.

Corresponding API in DUP:

```
VOID NamedValue::setValue(CONST INTEGER namedvaluePtr,
CONST IA5String name, CONST IA5String value)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int ptr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String name	CONST IA5String name	The full name (including the full path) of the entry to be set with a value.	In	No change
String value	CONST IA5String value	The value	In	No change



Returned Value:

None

Note: If the second argument `name` is "", the value of the first level entry of the `NamedValue` structure is set.

Exception:

`FDSStandardException` is thrown if the `valuenam` cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "");
NamedValue.addValue(ptr, "II.III.IIII", "VALUE");
NamedValue.print(ptr); // Printout 1
NamedValue.setValue(ptr, "II.III.IIII", "value");
NamedValue.print(ptr); // Printout 2
NamedValue.delete(ptr);

Printout 1:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>VALUE</IIII>
    </III>
  </II>
</respList>

Printout 2:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>value</IIII>
    </III>
  </II>
</respList>
```

Example 37 setValue

6.8 removeValue

Prototype:

```
public static void removeValue(int namedvaluePtr, int
targetPtr) throws FDSStandardException
```

Description:

This function removes entries from the `NamedValue` structure.

Corresponding API in DUP:

```
VOID NamedValue::removeValue(CONST INTEGER namedvaluePtr,
CONST INTEGER targetPtr)
```

**Function Change:**

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
int targetPtr	CONST INTEGER targetPtr	The pointer of the entry to be removed.	In	No change

Returned Value:

None

Note: Only when `targetPtr` is the direct sub-entry of `namedvaluePtr`, the API function takes effect.

Exception:

`FDSStandardException` is thrown if the `valuenam` cannot be found.



```

int ptr1 = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr1, "II", "");
NamedValue.addValue(ptr1, "II.III", "");
NamedValue.addValue(ptr1, "II.III.IIII", "value");
int ptr2 = NamedValue.newNamedValue("JJ", "");
int ptr3 = NamedValue.newNamedValue("JJJ", "value");
NamedValue.appendNamedValue(ptr1, ptr2);
NamedValue.appendNamedValue(ptr2, ptr3);
NamedValue.print(ptr1); //Printout 1
NamedValue.removeValue(ptr1, ptr2);
NamedValue.print(ptr1); //Printout 2
NamedValue.delete(ptr3);
NamedValue.delete(ptr2);
NamedValue.delete(ptr1);

Printout 1:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>value</IIII>
    </III>
  </II>
  <JJ>
    <JJJ>value</JJJ>
  </JJ>
</respList>

Printout 2:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>value</IIII>
    </III>
  </II>
  <JJ></JJ>
</respList>

```

Example 38 removeValue

6.9 getName

Prototype:

```
public static String getName(int namedvaluePtr) throws
FDSStandardException
```

Description:

This function gets the tag name of the first level entry of the NamedValue structure.

Corresponding API in DUP:

```
IA5String NamedValue::getName(CONST INTEGER namedvaluePtr)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change

Returned Value:

The name of the entry.

Type: String

Exception:

FDSStandardException is thrown if the oldname cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");  
String name = NamedValue.getName(ptr); //name is "respList"
```

Example 39 *getName*

6.10 setName

Prototype:

```
public static void setName(int namedvaluePtr, String  
oldname, String newname) throws FDSStandardException
```

Description:

This function sets the tag name of an entry of the NamedValue structure.

Corresponding API in DUP:

```
VOID NamedValue::setName(CONST INTEGER namedvaluePtr,  
CONST IA5String oldname, CONST IA5String newname)
```

Function Change:

No other change in function behavior.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String oldname	CONST IA5String oldname	The name of the entry (including the full path) to be changed.	In	No change
String newname	CONST IA5String newname	The new tag name.	In	No change

Returned Value:

None

Note: If the second parameter `oldname` is set to "", the tag name of the first level entry is changed.

Exception:

`FDSStandardException` is thrown if the `oldname` cannot be found.

```
int ptr = NamedValue.newNamedValue("I", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.JJJ", "");
NamedValue.addValue(ptr, "II.JJJ.IIII", "value");
NamedValue.print(ptr); //Printout 1
NamedValue.setName(ptr, "II.JJJ", "III");
NamedValue.setName(ptr, "", "respList");
NamedValue.print(ptr); //Printout 2
NamedValue.delete(ptr);

Printout 1:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<I>
  <II>
    <JJJ>
      <IIII>value</IIII>
    </JJJ>
  </II>
</I>

Printout 2:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>value</IIII>
    </III>
  </II>
</respList>
```

Example 40 `setName`



6.11 addAttribute

Prototype:

```
public static void addAttribute(int namedvaluePtr, String  
attributeName, String value) throws FDSStandardException
```

Description:

This function adds an attribute into an entry of the NamedValue structure.

Corresponding API in DUP:

```
VOID NamedValue::addAttribute(CONST INTEGER namedvaluePtr,  
CONST IA5String attributeName, CONST IA5String  
attributeValue)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String attributeName	CONST IA5String attributeName	The attribute name (including the full path).	In	No change
String value	CONST IA5String attributeValue	The attribute value.	In	No change

Returned Value:

None

Note: If the second parameter `oldname` is set to "", the tag name of the first level entry is changed.

Exception:

If the second argument `attributeName` is not a valid path or name, an error is reported.



```

int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "value");
NamedValue.addAttribute(ptr, "II.III.attribute", "3.0");
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III attribute="3.0">value</III>
  </II>
</respList>

```

Example 41 addAttribute

6.12 getAttribute

Prototype:

```
public static String getAttribute(int namedvaluePtr,
String attributeName) throws FDSStandardException
```

Description:

This function gets the attribute value of an entry of the NamedValue structure.

Corresponding API in DUP:

```
IA5String NamedValue::getAttribute(CONST INTEGER
namedvaluePtr, CONST IA5String attributeName)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String attributeName	CONST IA5String attributeName	The attribute name (including the full path).	In	No change

Returned Value:

The attribute value.



Type: String

Note: If the second argument `attributeName` is not a valid path or name, an error is reported.

Exception:

`FDSStandardException` is thrown if the second argument `attributeName` is not a valid path or name.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addAttribute(ptr, "imsi", "22223333");
String str = NamedValue.getAttribute(ptr, "imsi"); //str is "22223333"
NamedValue.delete(ptr);
```

Example 42 `getAttribute`

6.13 `setAttribute`

Prototype:

```
public static void setAttribute(int namedvaluePtr, String
attributeName, String value) throws FDSStandardException
```

Description:

This function sets the attribute value of an entry of the `NamedValue` structure.

Corresponding API in DUP:

```
VOID NamedValue::setAttribute(CONST INTEGER namedvaluePtr,
CONST IA5String attributeName, CONST IA5String
attributeValue)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
<code>int namedvaluePtr</code>	<code>CONST INTEGER namedvaluePtr</code>	The pointer of the <code>NamedValue</code> structure.	In	No change



Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String attribute Name	CONST IA5String attributeName	The attribute name (including the full path).	In	No change
String value	CONST IA5String attributeValue	The attribute value	In	No change

Returned Value:

None.

Note: If the second argument `attributeName` is only an attribute name (excluding the full path), the attribute value of the first level entry is reset.

Exception:

`FDSStandardException` is thrown if the second argument `attributeName` is not a valid path or name.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addAttribute(ptr, "attribute", "3.0");
NamedValue.print(ptr);           //Printout 1
NamedValue.setAttribute(ptr, "attribute", "4.0");
NamedValue.print(ptr);           //Printout 2
NamedValue.delete(ptr);

Printout 1:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList attribute="3.0"></respList>

Printout 2:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList attribute="4.0"></respList>
```

Example 43 `setAttribute`

6.14 attributes

Prototype:

```
public static int attributes(int namedvaluePtr, String
valueName) throws FDSStandardException
```

Description:

This function gets the attribute number of an entry of the `NamedValue` structure.

**Corresponding API in DUP:**

```
INTEGER NamedValue::attributes(CONST INTEGER namedvaluePtr  
, CONST IA5String valueName)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String value	CONST IA5String valueName	The tag name (including the full path) of the entry from which the attribute number is received.	In	No change

Returned Value:

The attribute number.

Type: int

Exception:

FDSStandardException is thrown if the valueName cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");  
NamedValue.addValue(ptr, "II", "");  
NamedValue.addValue(ptr, "II.III", "value");  
NamedValue.addAttribute(ptr, "imsi", "22220000");  
NamedValue.addAttribute(ptr, "II.III.attribute", "1.0");  
int num = NamedValue.attributes(ptr, ""); // num is 1  
num = NamedValue.attributes(ptr, "II.III"); // num is 1  
num = NamedValue.attributes(ptr, "II"); // num is 0  
NamedValue.delete(ptr);
```

Example 44 attributes

6.15 getAttributeAt

Prototype:

```
public static int getAttributeAt(int ptr, String  
valueName, int position) throws FDSStandardException
```



Description:

This function gets the attribute value of an entry from a given position.

Corresponding API in DUP:

```
INTEGER NamedValue::getAttributeAt(CONST INTEGER
namedvaluePtr, CONST IA5String valueName, CONST INTEGER
position)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int ptr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String valueName	CONST IA5String valueName	The tag name (including the full path) of the entry from which the attribute value is received.	In	No change
int position	CONST INTEGER position	The position of the attribute in the entry, counted from 0.	In	No change

Returned Value:

The identifier of the attribute entry.

Type: int

Note: If the second argument `valueName` is "", the attribute value is received from the first level entry.

Exception:

`FDSStandardException` is thrown if the `valueName` cannot be found, or the position is out of bound.



```
int ptr1 = NamedValue.newNamedValue("respList", "");
NamedValue.addAttribute(ptr1, "a1", "1.0");
NamedValue.addAttribute(ptr1, "a2", "2.0");
int attr = NamedValue.getAttributeAt(ptr1, "", 1);
NamedValue.print(attr);
NamedValue.delete(ptr1);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<a2>2.0</a2>
```

Example 45 *getAttributeAt*

6.16 addDu

Prototype:

```
public static void addDu(int namedValuePtr, String
valueName, Object du) throws FDSStandardException
```

Description:

This function encodes a Java Bean object to the `NamedValue` structure and adds it into another given `NamedValue` structure.

Corresponding API in DUP:

```
VOID NamedValue::addDu(CONST INTEGER namedvaluePtr, CONST
IA5String valueName, CONST ANY Du)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedValuePtr	CONST INTEGER namedvaluePtr	The pointer of the <code>NamedValue</code> structure into which an ASN.1 type is added.	In	No change
String valueName	CONST IA5String valueName	The tag name (including the full path) of the entry into which the ASN.1 type is added.	In	No change
Object du	CONST ANY Du	The Java Bean object.	In	No change



Returned Value:

None.

Note: If the second argument `valueName` is "", the attribute value is received from the first level entry.

Exception:

`FDSStandardException` is thrown if the `valueName` cannot be found.

```
public class SubscriptionExample{
    protected Integer t11;

    public Integer getT11() {
        return t11;
    }

    public void setT11(Integer value) {
        this.t11 = value;
    }
}

SubscriptionExample sub = new SubscriptionExample();
sub.setT11(23);
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "");
NamedValue.addDu(ptr, "II.III", sub);
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <SubscriptionExample>
        <t11>23</t11>
      </SubscriptionExample>
    </III>
  </II>
</respList>
```

Example 46 addDu

6.17 get

Prototype:

```
public static int get(int ptr, String name) throws
FDSStandardException
```

Description:

This function gets an entry and its sub-entry from the `NamedValue` structure.

**Corresponding API in DUP:**

```
INTEGER NamedValue::get(CONST INTEGER namedvaluePtr, CONST  
IA5String valueName)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int ptr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String name	CONST IA5String valueName	The tag name (including the full path) of the entry to be got.	In	No change

Returned Value:

The pointer of the entry with its sub-entries.

Note: If the second parameter valueName is "", the whole NamedValue structure is retrieved.

Exception:

FDSStandardException is thrown if the valueName cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");  
NamedValue.addValue(ptr, "II", "");  
NamedValue.addValue(ptr, "II.III", "");  
NamedValue.addValue(ptr, "II.III.IIII", "value");  
int subPtr = NamedValue.get(ptr, "II.III");  
NamedValue.print(subPtr);  
NamedValue.delete(ptr);  
  
Printout:  
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>  
<III>  
  <IIII>value</IIII>  
</III>
```

Example 47 *get*



6.18 getWithAttributes

Prototype:

```
public static int getWithAttributes(int ptr, String
valueName, ArrayList<Integer> attributes) throws
FDSStandardException
```

Description:

This function gets an entry due to the comparison of its attributes.

Corresponding API in DUP:

```
INTEGER NamedValue::getWithAttributes(CONST INTEGER
namedvaluePtr, CONST IA5String valueName, CONST ANY list)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int ptr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String name	CONST IA5String valueName	The tag name (including the full path) of the entry to be got.	In	No change
ArrayList<Integer> attributes	CONST ANY list	A list storing some variables to be compared with the attributes of the entry specified by the second parameter.	In	No change

Returned Value:

The pointer of the entry to be received.

Type: int

Exception:

FDSStandardException is thrown if no attributes are provided or the valueName is invalid

```
int ptr1 = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr1, "II", "");
NamedValue.addValue(ptr1, "II.III", "");
NamedValue.addValue(ptr1, "II.III.IIII", "VALUE");
NamedValue.addAttribute(ptr1, "a1", "1.0");
NamedValue.addAttribute(ptr1, "a2", "2.0");
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(0,NamedValue.getAttributeAt(ptr1,"",1));
list.add(1,NamedValue.getAttributeAt(ptr1,"",0));

int ptr3 = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr3, "II", "");
NamedValue.addValue(ptr3, "II.III", "");
NamedValue.addValue(ptr3, "II.III.IIII", "value");
int ptr4 = NamedValue.get(ptr3,"II");
NamedValue.addAttribute(ptr4, "a1", "1.0");
NamedValue.addAttribute(ptr4, "a2", "2.0");
int ptr2 = NamedValue.getWithAttributes(ptr3,"II",list);
NamedValue.print(ptr2);
NamedValue.delete(ptr4);
NamedValue.delete(ptr3);
NamedValue.delete(ptr1);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<II a1="1.0" a2="2.0">
  <III>
    <IIII>value</IIII>
  </III>
</II>
```

Example 48 getWithAttributes

6.19 entries

Prototype:

```
public static int entries(int namedvaluePtr, String
valueName) throws FDSStandardException
```

Description:

This function gets the number of the sub-entries of a specific entry with its value name.

Corresponding API in DUP:

```
INTEGER NamedValue::entries(CONST INTEGER namedvaluePtr,
CONST IA5String valueName)
```

Function Change:

No other change in function behavior.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String valueName	CONST IA5String valueName	The value name (including the full path) of the entry.	In	No change

Returned Value:

The number of the sub-entries.

Type: int

Exception:

FDSStandardException is thrown if the valueName cannot be found.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "value");
NamedValue.addValue(ptr, "II.ac", "ac");
NamedValue.addValue(ptr, "II.sd", "232");
int num = NamedValue.entries(ptr, "II"); //num is 3
NamedValue.delete(ptr);
```

Example 49 entries

6.20

getAt

Prototype:

```
public static int getAt(int namedvaluePtr, String
valueName, int position) throws FDSStandardException
```

Description:

This function gets the sub_entry by a given position.

Corresponding API in DUP:

```
INTEGER NamedValue::getAt(CONST INTEGER namedvaluePtr,
CONST IA5String valueName, CONST INTEGER position)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure.	In	No change
String valueName	CONST IA5String valueName	The value name (including the full path) of the entry from which the sub-entry is received.	In	No change
int position	CONST INTEGER position	The index number of the sub-entry, counted from 0.	In	No change

Returned Value:

The pointer of the sub-entry.

Type: int

Note: If the second parameter valueName is "", the sub-entry is retrieved from the root (first level).

Exception:

FDSStandardException is thrown if the valueName cannot be found or the position is out of bound.

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "");
NamedValue.addValue(ptr, "II.III.IIII", "value");
NamedValue.addValue(ptr, "II.III.IIII", "");
NamedValue.addValue(ptr, "_II", "");

int subPtr = NamedValue.getAt(ptr, "", 1);
NamedValue.print(subPtr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<_II></_II>
```

Example 50 *getAt*

6.21 convert2ASN1

Prototype:

```
public static void convert2ASN1(int namedValuePtr, Object
du) throws FDSStandardException
```

**Description:**

This function converts a NamedValue structure to a Java Bean object.

Corresponding API in DUP:

```
VOID NamedValue::convert2ASN1(CONST INTEGER namedvaluePtr,  
CONST ANY du)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int namedvaluePtr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure to be converted.	In	No change
Object du	CONST ANY du	The Java Bean object to store the converted NamedValue structure.	In/Out	No change

Returned Value:

None

Exception:

FDSStandardException is thrown if the Java Bean Object does not match the NamedValue structure.



```
public class SubscriptionExample{
    protected Integer t11;

    public Integer getT11() {
        return t11;
    }

    public void setT11(Integer value) {
        this.t11 = value;
    }
}

int ptr = NamedValue.newNamedValue("TestSubscription", "");
NamedValue.addValue(ptr, "t11", "12");
SubscriptionExample sub = new SubscriptionExample();
Integer tmp = sub.getT11(); //tmp is null
NamedValue.convert2ASN1(ptr, sub);
tmp = sub.getT11(); //tmp is 12
NamedValue.delete(ptr);
```

Example 51 convert2ASN1

6.22 convert2Nv

Prototype:

```
public static int convert2Nv(Object du)
```

Description:

This function converts a Java Bean object to a NamedValue structure.

Corresponding API in DUP:

```
INTEGER NamedValue::convert2Nv(CONST ANY du)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Object du	CONST ANY du	The Java Bean object data to be converted.	In	No change

Returned Value:

The pointer of the NamedValue structure that stores the converted Java Bean object data.



Type: int

```
public class SubscriptionExample{
    protected Integer t11;

    public Integer getT11() {
        return t11;
    }

    public void setT11(Integer value) {
        this.t11 = value;
    }
}

SubscriptionExample sub = new SubscriptionExample();
sub.setT11(12);
int ptr = NamedValue.convert2Nv(sub);
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<SubscriptionExample>
  <t11>12</t11>
</SubscriptionExample>
```

Example 52 convert2Nv

6.23 buildFromString

Prototype:

```
public static int buildFromString(String string, String
option) throws FDSStandardException
```

```
public static int buildFromString(String string)
```

Description:

This function builds a NamedValue structure from a string.

Corresponding API in DUP:

```
INTEGER NamedValue::buildFromString(CONST IA5STRING
string, CONST IA5STRING option)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String string	CONST IA5STRING string	String	In	No change
String option	CONST IA5STRING option	String, type strip_ns to strip all namespace related attributes and prefixes.	In	No change

Returned Value:

The NamedValue identifier.

Type: int

Note: If an attribute of the Java Bean data has no value, the converted NamedValue structure does not contain the attribute.

Exception:

FDSStandardException is thrown if creating NamedValue fails.

```
String str = "<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>"+
    "<respList imsi=\"22220000\">"+
    "  <II>"+
    "    <III attribute=\"1.0\">value</III>"+
    "  </II>"+
    "  <ii>"+
    "    <iii>123</iii>"+
    "  </ii>"+
    "</respList>";
int ptr = NamedValue.buildFromString(str);
NamedValue.print(ptr);
NamedValue.delete(ptr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList imsi="22220000">
  <II>
    <III attribute="1.0">value</III>
  </II>
  <ii>
    <iii>123</iii>
  </ii>
</respList>
```

Example 53 *buildFromString*

6.24 convert2String

Prototype:

```
public static String convert2String(int ptr)
```



Description:

This function converts a NamedValue structure to a string.

Corresponding API in DUP:

```
IA5String NamedValue::convert2String(CONST INTEGER
namedvaluePtr)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int ptr	CONST INTEGER namedvaluePtr	The pointer of the NamedValue structure to be converted.	In	No change

Returned Value:

The string converted from the NamedValue structure.

Type: String

```
int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "");
NamedValue.addValue(ptr, "II.III.IIII", "value");
String strPtr = NamedValue.convert2String(ptr);
NamedValue.delete(ptr);

strPtr printout is :

<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>
      <IIII>value</IIII>
    </III>
  </II>
</respList>
```

Example 54 convert2String



6.25 appendNamedValue

Prototype:

```
public static void appendNamedValue(int targetNV, int  
sourceNV)
```

Description:

This function appends a NamedValue structure to another NamedValue structure.

Corresponding API in DUP:

```
VOID NamedValue::appendNamedValue(CONST INTEGER targetNV,  
CONST INTEGER sourceNV)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int targetNV	CONST INTEGER targetNV	The pointer of the target NamedValue structure.	In	No change
int sourceNV	CONST INTEGER sourceNV	The pointer of the source NamedValue structure.	In	No change

Returned Value:

None



```

int ptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(ptr, "II", "");
NamedValue.addValue(ptr, "II.III", "value");

int subPtr = NamedValue.newNamedValue("ii", "");
NamedValue.addValue(subPtr, "iii", "123");

NamedValue.appendNamedValue(ptr, subPtr);
NamedValue.print(ptr);
NamedValue.delete(ptr);
NamedValue.delete(subPtr);

Printout:
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<respList>
  <II>
    <III>value</III>
  </II>
  <ii>
    <iii>123</iii>
  </ii>
</respList>

```

Example 55 *appendNamedValue*

6.26 equal

Prototype:

```
public static boolean equal(int a, int b) throws
FDSStandardException
```

Description:

This function compares two NamedValue structures.

Corresponding API in DUP:

```
BOOLEAN NamedValue::equal(CONST INTEGER a_NV, CONST
INTEGER b_NV)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
int a_NV	CONST INTEGER a_NV	The pointer of the NamedValue structure.	In	No change
int b_NV	CONST INTEGER b_NV	The pointer of the NamedValue structure.	In	No change

**Returned Value:**

The result of the comparison, `true` means equal, `false` means not equal.

Type: `boolean`

Exception:

`FDSStandardException` is thrown if `a` or `b` cannot be found.

```
int aptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(aptr, "II", "");
NamedValue.addValue(aptr, "II.III", "value");
NamedValue.addValue(aptr, "II.ac", "ac");
NamedValue.addValue(aptr, "II.sd", "232");
NamedValue.addAttribute(aptr, "II.III.attribute", "1.0");
NamedValue.addAttribute(aptr, "imsi", "22220000");
int bptr = NamedValue.newNamedValue("respList", "");
NamedValue.addValue(bptr, "II", "");
NamedValue.addValue(bptr, "II.III", "value");
NamedValue.addValue(bptr, "II.ac", "ac");
NamedValue.addValue(bptr, "II.sd", "232");
NamedValue.addAttribute(bptr, "II.III.attribute", "1.0");
NamedValue.addAttribute(bptr, "imsi", "22220000");
boolean isEq = NamedValue.equal(aptr, bptr); //isEq is true
NamedValue.delete(aptr);
NamedValue.delete(bptr);
```

Example 56 equal

7 ServiceLayer Class

7.1 process

Prototype:

- 1 `public static Object process(String operation, Object mo) throws FDSStandardException`
- 2 `public static Object process(String operation, Object mo, boolean withPayLoad) throws FDSStandardException`
- 3 `public static Object process(QName trigger, String operation, Object mo) throws FDSStandardException`
- 4 `public static Object process(QName trigger, String operation, Object mo, boolean withPayLoad) throws FDSStandardException`



- 5 `public static Object process(QName trigger, String operation, Object mo, RequestHandler requestHandler, ResponseHandler responseHandler) throws FDSStandardException`
- 6 `public static Object process(QName trigger, String operation, Object mo, boolean withPayload, RequestHandler requestHandler, ResponseHandler responseHandler) throws FDSStandardException`

Description:

This function initializes a processing request on an MO.

Prototype 1, 2 and 5 are used to process a MVNE JDV.

Prototype 3, 4 and 5 are used to process a standard JDV.

Corresponding API in DUP:

ANY Engine::process(CONST IA5String operation, CONST ANY mo, CONST ANY basemo, CONST ANY filter, CONST ANY scope)

Function Change:

In Java, only the parameters `operation` and `mo` are supported.

Java API `ServiceLayer.process` throws `FDSStandardException`.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String operation	CONST IA5String operation	The operation to be performed on the MO. The operation only includes Create, Delete, Get, and Set.	In	No change
Object mo	CONST ANY mo	The MO to be operated	In	No change
QName trigger	-	The trigger of the standard JDV.	In	-



Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
boolean withPayload	-	Whether to set the payload into the Get and Delete request. If set to false, the payload of the Get and Delete request is null. The default value is false.	In	-
RequestHandler requestHandler	-	The requestHandler is to process the marshalled XML payload by MO Javabeen before sending to the callee.	In	-
ResponseHandler responseHandler	-	The responseHandler is to process the XML payload returned from the callee before unmarshalling response to MO Javabeen.	In	-

Returned Value:

The response of corresponding operation.

```
public void main(String theOperation, Subscription theMo,
BaseObject theBaseObject, Filter theFilter, Scope theScope) throws FDSStandardException {
...
Subscription sub = (Subscription) ServiceLayer.process("Get", theMo);
//process a get operation and return the result java bean
...
}
```

Example 57 process MVNE JDV

```
public void main(String theOperation, Subscription theMo,
BaseObject theBaseObject, Filter theFilter, Scope theScope) throws FDSStandardException {
...
QName qname = new QName("http://schemas.ericsson.com/ema/UserProvisioning/GsmHlr/", "GetSubscription");
hlrMoRef = (com.ericsson.pg.ca.gsmhler.GsmHlr.HlrSubscription) DU.cloneDU(ServiceLayer.process(qname, "Get", hlrMo));
//process a get operation and return the result java bean
...
}
```

Example 58 process Standard JDV

7.2 lookup

Prototype:

```
public static <T> void lookup(T theMo, List<ElementType>
dests) throws FDSStandardException
```




Description:

This function looks up the destination NEs for a specific MO in the lookup table.

Corresponding API in DUP:

```
VOID Engine::lookup(CONST ANY mo, ANY destinationElements)
```

Function Change:

Java API `ServiceLayer.lookup` throws `FDSStandardException` and returns NE instance name in Dynamic Activation.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
<code>T theMo</code>	<code>CONST ANY mo</code>	The MO to be used to find destination	In	No change
<code>List<ElementType> dests</code>	<code>ANY destinationElements</code>	The destination elements found in the lookup table.	Out	No change

Returned Value:

The response of corresponding operation.

```
ServiceLayer.lookup(theMo, dests); //find the routing according to theMo, and stored in dests
if(dests.size() < 1){
    throw new FDSStandardException("Failed to find routing");
}
```

Example 59 lookup

8 ResourceLayer Class

8.1 doNeOperation

Prototype:

```
public static void doNeOperation(String mmlCmd, Object mo,
    ElementType theDest) throws FDSStandardException
```

**Description:**

This function performs some operations towards the NE with *<NeName>*.

Corresponding API in DUP:

```
VOID MML::doNeOperation(CONST IA5STRING operation, CONST  
ANY operationData, CONST IA5STRING NeName)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String mmlCmd	CONST IA5STRING operation	The operation to be performed on the NE.	In	No change
Object mo	CONST ANY operationData	The data needed for the NE operation.	In	No change
ElementType theDest	CONST IA5STRING NeName	The NE to enable the operation.	In	No change

Returned Value:

None

```
public void main(String theOperation, Subscription theMo, BaseObject  
theBaseObject, Filter theFilter, Scope theScope) throws FDSStandardException{  
    ...  
    List<ElementType> theDest = new ArrayList<>();  
    ServiceLayer.lookup(theMo, theDest);  
    if (sizeOf(theDest) == 1){  
        ResourceLayer.doNeOperation("NokiaHlrNL:createAucSubscriber", theMo, theDest.get(0).getName());  
    }  
    ...  
}
```

Example 60 doNeOperation

8.2 doNeRetrieveOperation

Prototype:

```
public static <T> T doNeRetrieveOperation(String mmlCmd,  
Object mo, Object template, T response, String theDest)  
throws FDSStandardException
```



Description:

This function retrieves data in the given NE.

Corresponding API in DUP:

```
VOID MML::doNeRetrieveOperation(CONST IA5STRING
operation, CONST ANY filter, CONST ANY wantedParameters,
ANY response, CONST IA5STRING NeName, CONST ANY
wantedFilterAttr)
```

Function Change:

The parameter `response` of the `doNeRetrieveOperation` method in DUP can be changed after the method is called, but in Java the parameter can only be retrieved through the result value.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String <code>mmlCmd</code>	CONST IA5STRING <code>operation</code>	The operation to be performed on the NE.	In	No change
Object <code>mo</code>	CONST ANY <code>filter</code>	The data for the NE retrieve operation.	In	No change
Object <code>template</code>	CONST ANY <code>wantedParameters</code>	The wanted data from the NE printout.	In	No change
T <code>response</code>	ANY <code>response</code>	The response filled with wanted data.	In	The method parameter in DUP can be IN and OUT, while the parameter in Java is IN only.
String <code>theDest</code>	CONST IA5STRING <code>NeName</code>	The NE to enable the operation.	In	No change

Returned Value:

The result of `doNeRetrieveOperation`.



```
public Subscription main(String theOperation, Subscription theMo, BaseObject
theBaseObject, Filter theFilter, Scope theScope) throws FDSStandardException{
    ...
    List<ElementType> theDest = new ArrayList<>();
    Subscription theResponse;
    ServiceLayer.lookup(theMo, theDest);
    if (sizeof(theDest) == 1) {
        theResponse = ResourceLayer.doNeRetrieveOperation("ExampleNL::getFromOtherNamedValue",
theMo, null, theResponse, dests.get(0).getName());
    }
    ...
}
```

Example 61 doNeRetrieveOperation

9 ResourceLayerGenerator Class

9.1 getFirstNamedValue

Prototype:

```
public static String[] getFirstNamedValue()
public static String getParameterName()
public static String getParameterValue()
```

Description:

This function gets the first value in current value list.

Corresponding API in DUP:

```
VOID MmlGenerator::getFirstNamedValue(IA5STRING name,
IA5STRING value)
```

Function Change:

The first name and value are retrieved when the `getFirstNamedValue()` method in DUP is called

In Java the first name is retrieved only when the `getParameterName()` method is called. The first value is retrieved only when the `getParameterValue()` method is called.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
None	IA5STRING name	None	None	Currently, Java API does not support this parameter.
None	IA5STRING value	None	None	Currently, Java API does not support this parameter.

Returned Value:

For `public static String getParameterName()`, the returned value is the parameter name.

For `public static String getParameterValue()`, the returned value is the parameter value.

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='no'?>
<DU publicId="sip:4682200199@tcexpo.ics.se">
  <sip-trunking-control>
    <operator-configuration>
      <activated>true</activated>
      <disable-identity-validation></disable-identity-validation>
      <auxiliary-identity>sip:test@ericsson.com</auxiliary-identity>
      <static-route id="String">
        <id>String</id>
        <disabled></disabled>
        <standby-route></standby-route>
      </static-route>
    </operator-configuration>
  </sip-trunking-control>
</DU>

ResourceLayerGenerator.getFirstNamedValue();
ResourceLayerGenerator.getParameterName(); //return is "publicId"
ResourceLayerGenerator.getParameterValue(); //return is "sip:4682200199@tcexpo.ics.se"

ResourceLayerGenerator.getFirstSubNamedValue("publicId");
ResourceLayerGenerator.getParameterName(); //return is "publicId.sip-trunking-control"
ResourceLayerGenerator.getParameterValue(); //return is ""

ResourceLayerGenerator.getFirstSubNamedValue("sip-trunking-control");
ResourceLayerGenerator.getParameterName(); //return is "sip-trunking-control.operator-configuration"
ResourceLayerGenerator.getParameterValue(); //return is ""

ResourceLayerGenerator.getFirstSubNamedValue("operator-configuration");
ResourceLayerGenerator.getParameterName(); //return is "operator-configuration.activated"
ResourceLayerGenerator.getParameterValue(); //return is "true"

ResourceLayerGenerator.getNextSubNamedValue("operator-configuration");
ResourceLayerGenerator.getParameterName(); //return is "operator-configuration.disable-identity-validation"
ResourceLayerGenerator.getParameterValue(); //return is ""

ResourceLayerGenerator.getNextNamedValue();
ResourceLayerGenerator.getParameterName(); //return is "auxiliary-identity"
ResourceLayerGenerator.getParameterValue(); //return is "sip:test@ericsson.com"

ResourceLayerGenerator.getValueWithName("id"); //return is "String"
```

Example 62 `getFirstNamedValue`



9.2 getFirstSubNamedValue

Prototype:

```
public static String[] getFirstSubNamedValue(String name)
public static String getParameterName()
public static String getParameterValue()
```

Description:

This function gets the first value in the current sub value list.

Corresponding API in DUP:

```
VOID MmlGenerator::getFirstSubNamedValue(CONST IA5STRING
name, IA5STRING subname, IA5STRING subvalue)
```

Function Change:

The first subname and subvalue are retrieved when the `getFirstSubNamedValue()` method in DUP is called.

In Java the first subname is retrieved only when the `getParameterName()` method is called. The first subvalue is retrieved only when the `getParameterValue()` method is called.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5STRING name	The parameter name	In	No change

Returned Value:

For `public static String getParameterName()`, the returned value is the name of the sub-parameter.

For `public static String getParameterValue()`, the returned value is the value of the sub-parameter.

For details on the example of this method, see Example 62.



9.3 getNextNamedValue

Prototype:

```
public static String[] getNextNamedValue()  
public static String getParameterName()  
public static String getParameterValue()
```

Description:

This function gets the next value in current value list.

Corresponding API in DUP:

```
VOID MmlGenerator::getNextNamedValue(IA5STRING name,  
IA5STRING value)
```

Function Change:

The next name and value are retrieved when the `getNextNamedValue()` method in DUP is called.

In Java the next name is retrieved only when the `getParameterName()` method is called. The next value is retrieved only when the `getParameterValue()` method is called.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
None	IA5STRING name	None	None	Currently, Java API does not support this parameter.
None	IA5STRING value	None	None	Currently, Java API does not support this parameter.

Returned Value:

The next value in the current value list.

For details on the example of this method, see Example 62.



9.4 getNextSubNamedValue

Prototype:

```
public static String[] getNextSubNamedValue(String name)
public static String getParameterName()
public static String getParameterValue()
```

Description:

This function gets the next value in the current sub value list.

Corresponding API in DUP:

```
VOID MmlGenerator::getNextSubNamedValue(CONST IA5STRING
name, IA5STRING subname, IA5STRING subvalue)
```

Function Change:

The next subname and subvalue are retrieved when the `getNextSubNamedValue()` method in DUP is called.

In Java the next subname is retrieved only when the `getParameterName()` method is called. The next subvalue is retrieved only when the `getParameterValue()` method is called.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5STRING name	The parameter name	In	No change

Returned Value:

For `public static String getParameterName()`, the returned value is the name of the next value in the current sub-value list.

For `public static String getParameterValue()`, the returned value is the next value in the current sub-value list.

For details on the example of this method, see Example 62.



9.5 getValueWithName

Prototype:

```
public static String getValueWithName(String name)
```

Description:

This function gets a value from the current value list that is stored in Resource Layer Context.

Corresponding API in DUP:

```
IA5STRING MmlGenerator::getValueWithName(CONST IA5STRING  
name)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5STRIN G name	The parameter name.	In	No change

Returned Value:

The value corresponding to the parameter name.

For details on the example of this method, see Example 62.



10 ResourceLayerParser Class

10.1 setValue

Prototype:

```
public static void setValue(String variable, String value,  
String nullIsValid) throws FDSStandardException
```

Description:

This function sets the value of an entry that is stored in Resource Layer Context.

Corresponding API in DUP:

```
VOID MmlParser::setValue(CONST IA5STRING name, CONST  
IA5STRING value, CONST IA5STRING nullIsValid)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String variable	CONST IA5STRING name	The parameter name in the response list.	In	No change
String value	CONST IA5STRING value	The value of the parameter.	In	No change
String nullIsValid	CONST IA5STRING nullIsValid	0 (default): the null value is invalid; 1: the null value is valid.	In	No change

Returned Value:

None



```

ResourceContext.getRuntime().getOutputPayload(); //Printout 1
ResourceLayerParser.setValue("II", "122222222");
ResourceLayerParser.setValue("_II", "22333333");
ResourceContext.getRuntime().getOutputPayload(); //Printout 2

Printout 1:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<respList>
  <Create>
    <msisdn>33333333</msisdn>
  </Create>
</respList>

Printout 2:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<respList>
  <Create>
    <msisdn>33333333</msisdn>
    <II>122222222</II>
    <_II>22333333</_II>
  </Create>
</respList>

```

Example 63 *setValue*

10.2 setWholeNv

Prototype:

```
public static void setWholeNv(String theResponse)
```

Description:

This function returns a whole response list to ProcEngine.

Corresponding API in DUP:

```
VOID MmlParser::setWholeNv(CONST IA5String theResponse)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String theResponse	CONST IA5String theResponse	The whole response list. theResponse is in the NamedValue format and must match the corresponding MO definition.	In	No change

**Returned Value:**

None

```
int ptr = NamedValue.newNamedValue("TheResp", "");
NamedValue.addValue(ptr, "imsi", "");
NamedValue.addValue(ptr, "imsi.msisdn", "11110000");
NamedValue.print(ptr);

ResourceLayerParser.setWholeNv(NamedValue.convert2String(ptr));
ResourceContext.getRuntime().getOutputPayload();

Output:

<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<TheResp>
  <imsi>
    <msisdn>11110000</msisdn>
  </imsi>
</TheResp>
```

Example 64 setWholeNv

11 ConfigData Class

11.1 get

Prototype:

- 1 public static NamedValues get(String name,NamedValues out) throws FDSStandardException
- 2 public static String get(String name) throws FDSStandardException

Description:

Prototype 1 separates the configuration data with “,” and returns the customer data item as NamedValues.

Prototype 2 returns the configuration data as a string.

These two APIs are used in ServiceLayer JDV.

Corresponding API in DUP:

```
VOID ConfigData::get(CONST IA5STRING dataname, ANY
datavalue)
```



Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5STRIN G dataname	The data item has been registered.	In	No change
NamedValues out	ANY datavalue	The customer data item created in GUI.	Out	No change

Returned Value:

The customer data item.

Exception:

FDSStandardException if the customer data is not defined.

in *JavaDataViewFactory.java add the methods as follows:

```
@ConfigurationProperty(description = "Get RootDn")
public String getRootDn() {
    return configData.get("RootDn");
}
public void setRootDn(String value) {
    configData.put("RootDn", value);
}
```

Then configure it in GUI, for example, com=ericsson,ou=operation.

```
NamedValues out = new NamedValues();
out = ConfigData.get("RootDn", out);
//out is
//<NamedValues name="RootDn">
//<values value="com=ericsson"/>
//<values value="ou=operation"/>
//</NamedValues>
```

```
String str = ConfigData.get("RootDn"); // str is "com=ericsson,ou=operation"
```

Example 65 *get*



12 ConfigDataResourceLayer Class

12.1 get

Prototype:

- 1 `public static NamedValues get(String name,NamedValues out) throws FDSStandardException`
- 2 `public static String get(String name) throws FDSStandardException`

Description:

Prototype 1 separates the configuration data with “,” and returns the customer data item as `NamedValues`.

Prototype 2 returns the configuration data as a string.

These two APIs are used in ResourceLayer JDV.

Corresponding API in DUP:

```
VOID ConfigData::get(CONST IA5STRING dataname, ANY datavalue)
```

Function Change:

No other change in function behavior.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String name	CONST IA5STRING dataname	The data item has been registered.	In	No change
NamedValues out	ANY datavalue	The customer data item created in GUI.	Out	No change

Returned Value:

The customer data item.



Exception:

FDSStandardException if the customer data is not defined.

```
in *JavaDataViewFactory.java add the methods as follows:
@ConfigurationProperty(description = "Get RootDn")
public String getRootDn() {
    return configData.get("RootDn");
}
public void setRootDn(String value) {
    configData.put("RootDn", value);
}
```

Then configure it in GUI, for example, com=ericsson,ou=operation.

```
NamedValues out = new NamedValues();
```

```
    out = ConfigDataResourceLayer.get("RootDn", out);
```

```
//out is
```

```
//<NamedValues name="RootDn">
```

```
//<values value="com=ericsson"/>
```

```
//<values value="ou=operation"/>
```

```
//</NamedValues>
```

```
String str = ConfigDataResourceLayer.get("RootDn"); // str is "com=ericsson,ou=operation"
```

Example 66 get

13 Error Class

13.1 setCaiError

Prototype:

```
public static void setCaiError(FDSStandardException ex,
int code)
```

```
public static void setCaiError(FDSStandardException ex,
Object code) throws FDSStandardException
```

Description:

This function sets error code to FDSStandardException.

Corresponding API in DUP:

```
VOID Error::setCaiError(FDS.FDSStandardException
exception, CONST INTEGER caiError)
```

**Function Change:**

In DUP, `setCaiError` accepts both `Integer` and `Object` as the second parameter, therefore the second method is provided to support the `Object` input parameter.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
<code>FDSStandardException ex</code>	<code>FDS.FDSStandardException exception</code>	The exception	In	No change
Object code	<code>CONST INTEGER caiError</code>	The error code	In	No change

Returned Value:

None

```
Error.setErrorCode(111111); //set error code to 111111
Error.getCaiError(e); //get error code 111111

@XmlEnumValue("5")
idNotValid("5");
FDSStandardException e = new FDSStandardException();
Error.setCaiError(e, idNotValid);
int errorCode = Error.getCaiError(e); //get error code 5
```

Example 67 setCaiError

13.2 `getCaiError`

Prototype:

```
public static int getCaiError(FDSStandardException ex)
```

Description:

This function retrieves the CAI error code from `FDSStandardException` set by the API `setCaiError`.

Corresponding API in DUP:

```
INTEGER Error::getCaiError(FDS.FDSStandardException ex)
```

Function Change:

No other change in function behavior.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
FDSStandardException ex	FDS.FDSStandardException ex	The exception	In	No change

Returned Value:

CAI error codes.

Type: int

Note: Since the CAI error code is hidden in the hint attribute of the FDSStandardException, it cannot be directly retrieved.

```
FDSStandardException e = new FDSStandardException();
e.setErrorCode(111111);
int errorCode = Error.getCaiError(e); //get error code 111111
```

Example 68 *getCaiError*

13.3 externalError

Prototype:

```
Public static int externalError(String errorKey)
```

Description:

This function returns the error code of the specified external error.

Corresponding API in DUP:

```
INTEGER Error::externalError(CONST IA5String errorDescription)
```

Function Change:

No other change in function behavior.

**Parameter Table:**

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String errorKey	CONST IA5String errorDescription	The external error Mapping the error description in Table 3 to error code.	In	No change

Returned Value:

The error code mapped from the external error description.

Type: INTEGER

Table 3 shows relationship between the external error and error code:

Table 3 Mapping Table

External Error	Error Code
mainidalreadyexist	1
mainiddonotexist	2
idalreadyexist	3
iddonotexist	4
idnotvalid	5
paramorparamvalismissing	6
paramorparamvalnotvalid	7
physmaxofdataareached	8
logmaxofdataareached	9
datarecordcurrentlylocked	10
functionalitynotsupported	11
otherfault	99

```
FDSStandardException toThrow = new FDSStandardException();  
toThrow.setErrorCode(Error.externalError("functionalitynotsupported  
")); // 11 is set as error code
```

Example 69 *externalError*



14 FDSStandardException

`FDSStandardException` class is used for exceptions in the converted JDV project. It has the same parameters (reason, hint, and errorcode) behavior as those of DUP `FDSStandardException`.

14.1 setErrorCode

Prototype:

```
public void setErrorCode(Object errorCode) throws  
FDSStandardException
```

```
public void setErrorCode(int errorCode)
```

Description:

This function sets the error code of the `FDSStandardException`.

Parameter Table:

Parameter in Java	Description
Object errorCode	The error code of the <code>FDSStandardException</code> .
int errorCode	

Returned Value:

None

Exception:

For the first method, if the value of the Object `errorCode` cannot be found, `FDSStandardException` is thrown.



```
FDSStandardException e = new FDSStandardException();
    e.setErrorCode(1); //set error code with int
    e.setReason("IMSI ALREADY DEFINED");
    e.setHint("Please use another IMSI");
int errorCode = e.getErrorCode(); //error code is 1
String reason = e.getReason(); //reason is "IMSI ALREADY DEFINED"
String hint = e.getHint(); //hint is "Please use another IMSI"

public enum External {
    @XmlEnumValue("5")
    IDNOTVALID("5")
}
e.setErrorCode(External.IDNOTVALID); //set error code with object
int errorCode = e.getErrorCode(); //errorCode is 5
```

Example 70 setErrorCode

14.2 getErrorCode

Prototype:

```
public int getErrorCode()
```

Description:

This function gets the error code of the `FDSStandardException`.

Parameter Table:

None

Returned Value:

Int `errorCode`: the error code of the `FDSStandardException`.

For details, refer to Example 70.

14.3 setReason

Prototype:

```
public void setReason(String reason)
```

Description:

This function sets the reason of the `FDSStandardException`.

**Parameter Table:**

Parameter in Java	Description
String reason	The reason of the FDStandardException.

Returned Value:

None

For details, refer to Example 70.

14.4 getReason

Prototype:

```
public String getReason()
```

Description:

This function gets the reason of the FDStandardException.

Parameter Table:

None

Returned Value:

String reason: the reason of the FDStandardException.

For details, refer to Example 70.

14.5 setHint

Prototype:

```
public void setHint(String hint)
```

Description:

This function sets the hint of the FDStandardException.

**Parameter Table:**

Parameter in Java	Description
String hint	The hint of the <code>FDSStandardException</code> .

Returned Value:

None

For details, refer to Example 70.

14.6 getHint

Prototype:

```
public String getHint()
```

Description:

This function gets the hint of the `FDSStandardException`.

Parameter Table:

None

Returned Value:

String hint: the hint of the `FDSStandardException`.

For details, refer to Example 70.

14.7 toJDVException

Prototype:

```
public static JavaDataViewException toJDVException(FDSStandardException e)
```

Description:

Because JDVs only accept `JavaDataViewException`, this function is used to change a `FDSStandardException e` to a `JavaDataViewException`.

**Parameter Table:**

Parameter in Java	Description
FDSStandardException e	The FDSStandardException.

Returned Value:

JavaDataViewException e1: the switched JDVException.

```
FDSStandardException e = new FDSStandardException();
JavaDataViewException jdvE = FDSStandardException.toJDVException(e);
//the FDSStandardException e is changed to JavaDataViewException jdvE
```

Example 71 toJDVException

14.8 toFDSEException

Prototype:

```
public static FDSStandardException toFDSEException(JavaData
ViewException e)
```

Description:

This function changes a JavaDataViewException to a FDSStandardException e.

Parameter Table:

Parameter in Java	Description
JavaDataViewException e	The JavaDataViewException.

Returned Value:

FDSStandardException e1: the switched FDSStandardException.



```
FDSStandardException e = new FDSStandardException();
    e.setErrorCode(1);//set error code with int
    e.setReason("IMSI ALREADY DEFINED");
    e.setHint("Please use another IMSI");
    JavaDataViewException jdvE = FDSStandardException.toJDVException(e);
    int errorCode = jdvE.getAdditionalError().getErrorCode();//error code is 1
    String reason = jdvE.getAdditionalError().getErrorDescription();//reason is "IMSI ALREADY DEFINED"
    String hint = (String)jdvE.getAdditionalError().getUserData();//hint is "Please use another IMSI"

    FDSStandardException fdse = FDSStandardException.toFDSEException(jdvE);
// the JavaDataViewException jdvE is changed to FDSStandardException fdse
    errorCode = e.getErrorCode();//error code is 1
    reason = e.getReason();//reason is "IMSI ALREADY DEFINED"
    hint = e.getHint();//hint is "Please use another IMSI"
```

Example 72 *toFDSEException*

15 Misc Class

15.1 exIsDefined

Prototype:

```
public static boolean exIsDefined(Object mo, String
attrList) throws FDSStandardException
```

Description:

This function checks if any attribute in an attribute list has been assigned a value in a given MO.

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
Object mo	CONST ANY theMo	The MO	In	No change
String attrList	CONST IA5STRING attributelist	The attribute names list with "+" as delimiter between every two attributes.	In	No change

Returned Value:

Return true if any attribute in the attributelist has been assigned a value; Return false if no attribute has value.



Exception:

FDSStandardException if any attribute in attributelist is not included in theMo.

Note: The checking sequence for those attributes in attributelist is from left to right. true will be immediately returned when find the first attribute has value.

```
Subscription sub = new Subscription();
sub.setBs21(1);
String test = "acc+aoc+bs21+bs22+bs23";

Boolean result = Misc. exIsDefined(sub, test); // result is true
```

Example 73 *exIsDefined*

16 Build Class

16.1 convertToCai3g12Request

Prototype:

- `Public static String convertToCai3g12Request(String operation, String mOType, List<String> mOId, List<String> mOAttributes, String extension) throws FDSStandardException`
- `public static String convertToCai3g12Request(String operation, String mOType, List<String> mOId) throws FDSStandardException`

Description:

This function generates one CAI3G 1.2 format request according to the user-defined input.

**Note:**

- If `extension` is not expected, but `mOAttributes` is needed, the empty string (that is `""`) will be passed to the extension. If neither of them is needed, the first three parameters can be presented alone.
- The input of `mOld` and `mOAttributes` must be set in form of `name` and `value` pairs in sequence. `name` can be an attribute or an element with the value of the CAI3G request.
- In the generated CAI3G request, if `name` is set as an attribute instead of an element, it must be prefixed with `@`.
- If MO Attributes contains `SubobjectAttribute` or `StructureAttribute` in the generated CAI3G request, the path of `name` must be delimited by `"."` in order to identify the current attribute or element level from the MO attributes top level. `&` must be set as the prefix in the first declared parent element level in `name`. If no values are assigned to MO Attributes, the corresponding request will not have the element `<MOAttributes/>`.
- This API complies with CAI3G 1.2 standards, while `Build::convertToCai3gRequest` complies with 1.1.

Corresponding API in DUP:

```
IA5STRING Build::convertToCai3g12Request(CONST IA5STRING
operation, CONST IA5STRING mOType, CONST ANY mOId, CONST
ANY mOAttributes, CONST IA5STRING extension)
```

Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String operation	CONST IA5STRIN G operation	Identify the operation for the generated CAI3G request, possible values are Create, Set, Delete, and Get.	In	No change
String mOType	CONST IA5STRIN G mOType	The MO type of the generated CAI3G request.	In	No change
List<String> mOId	CONST ANY mOId	The list of identifiers for the MO of the generated CAI3G request.	In	No change



Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
List<String> moAttributes	CONST ANY moAttributes	The list of attributes for the MO of the generated CAI3G request (optional).	In	No change
String extension	CONST IA5STRIN G extension	The extension data of the request (optional)	In	No change

Returned Value:

The CAI3G format request.

If any invalid parameter appears, the `null` string will be returned.

Exception:

`FDSStandardException` is thrown if parameter `moAttributes` is not a valid named value.

```
public void testConvertGet3() throws Exception {
    String operation = "Get";
    String moType = " ";
    moType = "SubscriberData@http://schemas.ericsson.com/pg/hlr/13.5/";
    List<String> moId = new ArrayList<>();
    moId.add("imsi");
    moId.add("1234");
    moId.add("msisdn");
    moId.add("56677");

    List<String> moAttributes = new ArrayList<>();
    moAttributes.add("<rid>111</rid>");
    moAttributes.add("<mscid>222</mscid>");

    String resp = Build.convertToCai3g12Request(operation, moType, moId, moAttributes, null);
    logger.info(resp);
    CAI3GMessageWrapper wrapper = new CAI3GMessageWrapper(XMLUtil.getDocument(resp));
    logger.info(XMLUtil.toString(wrapper.toSoapRequestDocument()));
}
```

Example 74 *convertToCai3g12Request*

16.2 convertToCai3gRequest

Prototype:

- `public static String convertToCai3gRequest(String operation, String moType, List<String> moId, List<String> moAttributes, String extension) throws FDSStandardException`



- `public static String convertToCai3gRequest(String operation, String mOType, List<String> mOId) throws FDSStandardException`

Description:

This function generates one CAI3G format request according to the user-defined input. It has the same function as Section 16.1 on page 91.

Note:

- If `extension` is not expected, but `mOAttributes` is needed, the empty string (that is "") will be passed to the extension. If neither of them is needed, the first three parameters can be presented alone.
- The input of `mOId` and `mOAttributes` must be set in form of name and value pairs in sequence. `name` can be an attribute or an element with the value of the CAI3G request.
- In the generated CAI3G request, if `name` is set as an attribute instead of an element, it must be prefixed with @.
- If MO Attributes contains `SubobjectAttribute` or `StructureAttribute` in the generated CAI3G request, the path of `name` must be delimited by "." in order to identify the current attribute or element level from the MO attributes top level. & must be set as the prefix in the first declared parent element level in `name`. If no values are assigned to MO Attributes, the corresponding request will not have the element `<MOAttributes/>`.
- This API complies with CAI3G 1.2 standards, while `Build::convertToCai3gRequest` complies with 1.1.

Corresponding API in DUP:

```
IA5STRING Build::convertToCai3gRequest(CONST IA5STRING  
operation, CONST IA5STRING mOType, CONST ANY mOId, CONST  
ANY mOAttributes, CONST IA5STRING extension)
```

Function Change:

Do not support CAI3G1.1.



Parameter Table:

Parameter in Java	Corresponding Parameter in DUP	Description	In/Out	Change
String operation	CONST IA5STRING operation	Identify the operation for the generated CAI3G request, possible values are Create, Set, Delete, and Get.	In	No change
String mOType	CONST IA5STRING mOType	The MO type of the generated CAI3G request	In	No change
List<String> mOId	CONST ANY mOId	The list of identifiers for the MO of the generated CAI3G request	In	No change
List<String> mOAttributes	CONST ANY mOAttributes	The list of attributes for the MO of the generated CAI3G request (optional)	In	No change
String extension	CONST IA5STRING extension	The extension data of the request (optional)	In	No change

Returned Value:

The CAI3G format request.

If any invalid parameter appears, the `null` string will be returned.

Exception:

`FDSStandardException` is thrown if parameter `mOAttributes` is not a valid named value.



```
public void testConvertGet3() throws Exception {  
    String operation = "Get";  
    String moType = "";  
    moType = "SubscriberData@http://schemas.ericsson.com/pg/hlr/13.5/";  
    List<String> moId = new ArrayList<>();  
    moId.add("imsi");  
    moId.add("1234");  
    moId.add("msisdn");  
    moId.add("56677");  
  
    List<String> moAttributes = new ArrayList<>();  
    moAttributes.add("<rid>111</rid>");  
    moAttributes.add("<mscid>222</mscid>");  
  
    String resp = Build.convertToCai3g12Request(operation, moType, moId, moAttributes, null);  
    logger.info(resp);  
    CAI3GMessageWrapper wrapper = new CAI3GMessageWrapper(XMLUtil.getDocument(resp));  
    logger.info(XMLUtil.toString(wrapper.toSoapRequestDocument()));  
}
```

Example 75 convertToCai3gRequest



Reference List

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Multi Activation IDE Reference Manual*, 1/2134-CSH 109 434 Uen