

# Programmers Guide for Consistency Checker

Ericsson Dynamic Activation

---

## USER GUIDE

**Copyright**

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

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

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



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	Typographic Conventions	1
1.4	Prerequisites	2
<b>2</b>	<b>Deployment View</b>	<b>2</b>
<b>3</b>	<b>Architectural Principle</b>	<b>3</b>
3.1	Package Name Convention	3
3.2	System Properties Adaptations	3
3.3	Commons Logging	4
3.4	Log Levels	4
3.5	Easy Adaptations	5
3.6	Logical Dependencies	5
3.7	Producing Large Files	5
<b>4</b>	<b>Customization</b>	<b>6</b>
4.1	Creating a New Rule	6
4.1.1	Prerequisites	6
4.1.2	Implementing the Rule	6
4.1.3	Code Examples	7
4.1.4	Deploying the Rule	9
4.2	Creating a New Data Model	9
4.2.1	Creating a New Off-line Data Model	10
4.2.2	Creating a New Online Data Model	10
4.2.3	Deploying the New Off-line Data Model	11
4.2.4	Deploying the New Online Data Model	11
4.3	Creating a New Off-line Data Parser	12
4.3.1	Prerequisites	12
4.3.2	Implementing the New Data Parser	12
4.3.3	Code Examples	13
4.3.4	Deploying the New Data Parser	13
4.4	Generic Extractors	14
4.4.1	Deploy the Generic Extraction Module	14
4.4.2	Place an Extraction Order with the Generic SQL Extraction Handler	14
4.4.3	Place an Extraction Order with the Generic LDAP Extraction Handler	15



4.5	Online Data Source Integration	16
4.5.1	Online Architecture Overview	17
4.5.2	Integration Scenarios	17
4.5.3	Developing Resource Adapter	18
4.5.4	Online Data Source Definition	20
4.6	Notification	22
4.7	GUI Language	24
4.8	Excel Reports	25
4.8.1	Disable the Export of Excel Reports	26
4.8.2	Customize Excel Reports	26
<b>5</b>	<b>Definitions</b>	<b>28</b>
5.1	Dump File Format	28
5.2	Rule Based Analysis	29
5.2.1	Rule Based Analysis Result File Format	29
5.3	Pattern Based Analysis	33
5.3.1	Pattern Based Analysis Result File Format	33
	<b>Reference List</b>	<b>37</b>



# 1 Introduction

The Consistency Checker is a generic data comparison tool for telecom operators and is used for exploring data consistency status. The consistency checker compares arbitrary data from arbitrary data sources and presents the result in the form of report and result files.

## 1.1 Purpose and Scope

The purpose of this document is to give the reader guidelines and examples of how to customize the Consistency Checker.

## 1.2 Target Groups

For Ericsson™ Service Organization, please contact the Product Unit for access to the Consistency Checker community.

## 1.3 Typographic Conventions

The typographic conventions used in this document are shown in Table 1.

*Table 1 Typographic Conventions*

Convention	Description	Example
<b>Output Information</b>	Text displayed by the system is shown in monospaced font.	System awaiting input
<b>User Input</b>	A command that must be entered in a Command Line Interface (CLI) exactly as written is shown in bold monospaced font.	<b>cd \$HOME</b>
<b>Command Variables</b>	Command variables included in a command, are enclosed by angle brackets <>. They are shown in bold, italic monospaced font.	<b>&lt;home_directory&gt;</b>
<b>GUI Objects</b>	GUI objects, such as menus, fields, and buttons are shown in bold font.	Click <b>File &gt; Exit</b> .
<b>Key Combinations</b>	Key combinations are shown in bold font.  The plus sign (+) indicates that the keys must be pressed simultaneously.	Press <b>Ctrl+X</b> to delete the selected value.



Convention	Description	Example
<b>System Elements</b>	Command and parameter names, program names, path names, URLs, and directory names are shown in monospaced font.  Slash (/) is used throughout the Consistency Checker documentation. It might differ in practice depending on the target OS environment.	The files are located in <code>/etc/opt/ericsson</code>
<b>Code Examples</b>	Code examples are shown in monospaced font.  The backslash (\) is used to show where long lines are split.	<pre>private Map&lt;String, AttributeRule&gt;     attributeMap = \  new HashMap&lt;String, AttributeRule&gt;();</pre>

## 1.4 Prerequisites

The following are the prerequisites to make full use of this document:

- Basic knowledge about the Consistency Checker, see *Function Specification Consistency Checker*, Reference [2].
- Extensive Java™ programming knowledge and experience.

## 2 Deployment View

The Consistency Checker comes with the following enterprise applications:

- `APP_CC_Management.ear` – acts as a Controlling unit and is deployed as a standard enterprise application in a JEE application server and running inside the application server JVM process. Includes the package:

```
com.ericsson.consistency.controller
```

- `APP_Offline.ear` – acts as a Serving unit instance and is deployed as a standard enterprise application in a JEE application server and running inside the application server JVM process.
- `APP_Realtime.ear` – acts as a Serving unit instance and is deployed as a standard enterprise application in a JEE application server and running inside the application server JVM process.



At deployment time, all the customization implementations must be embedded in the enterprise applications, see Figure 1.

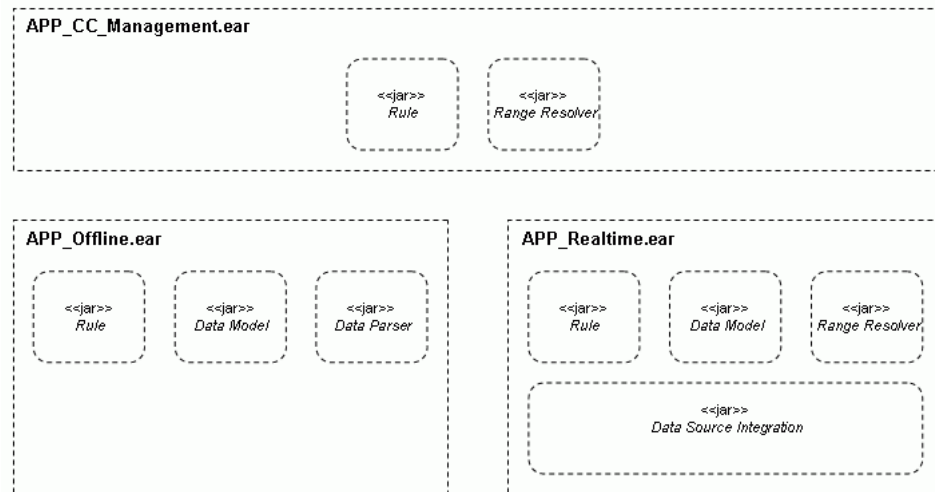


Figure 1 Deployment View of Customizations

Pattern based analysis does not have any customization points.

For further information about the Consistency Checker anatomy, see *System Administrators Guide for Consistency Checker*, Reference [1].

## 3 Architectural Principle

This section describes the architectural principle of the Consistency Checker.

### 3.1 Package Name Convention

The package name `com.ericsson.consistency` is reserved for the standard implementation. Customer adaptations or other extensions should not use this package name.

### 3.2 System Properties Adaptations

Java System properties are used as the configuration mechanism. The following naming convention is used for System properties keys in Consistency checker.



`<fully qualified class name>.<fieldname>`

Example:

```
com.ericsson.consistency.QueueHandler.MAX_QUEUE_SIZE
```

## 3.3 Commons Logging

To be independent of the underlying logging implementation the Apache™ Commons logging is used as logging proxy through out the whole Consistency Checker.

## 3.4 Log Levels

Log messages should always contain relevant dynamic information, with an explanation on the actually event. Preconceived consequences should be avoided, since it is not possible to know all future contexts the component will be used in.

Class names and method names should be avoided in the log message since that information is available in the underlying logger implementation, and have the possibilities to show this information if needed.

Debug and Trace levels messages should be surrounded by an if-statement.

Example:

```
if (logger.isDebugEnabled())
{
    logger.debug("Reading data from " +value);
}
```

*Table 2 Debug and Trace Messages*

Message	Description
Fatal	This level shall only be used when the problem is not recoverable. Example message: Fundamental prerequisites for processing an order are missing: Transaction ID.
Error	This level indicates that a major problem has occurred but the function that performs the actual logging is not capable to decide if the failure is recoverable or not.
Warning	This level shall be used to indicate that some parts of the task failed but it is possible to continue. Example message: Failed to read a line in the file c:/temp/myfile.txt.





Message	Description
Info	<p>The info level shall only be used for major process changes.</p> <p>Example message:</p> <p>The X-application is started by user ABC.</p>
Debug	<p>Debug level shall only display major processing decisions. It shall not in detail display the data that is processed.</p> <p>Example message:</p> <p>Loading libraries from directory c:/tmp/lib.</p>
Trace	<p>In this level it is allowed to display low level details.</p> <p>Example message:</p> <p>Comparing attribute ABC for record.</p>

## 3.5 Easy Adaptations

In all places where there might be a need for variation over the time, it is recommended to use the factory design pattern. In the Consistency Checker this pattern has slightly been extended with a quite common extra step. By applying this pattern the flexible architecture will remain. The pattern extension is about lookup its system property before the factory produces a new object. The system property should be optional and the value should be a fully qualified class name that should implement the interface that the factory produces. If the system property exists and it is possible to instantiate an object from its value that object should be returned from the factory `newInstance()` method.

## 3.6 Logical Dependencies

To avoid circular dependencies and potential dead locks, the Controller have knowledge about the serving units and not the other way around. Serving units should only call back via stored data, such as reports and results.

## 3.7 Producing Large Files

When large files are produced, for example, when an extraction handler collects and transforms data from data sources, there might be consuming processes that starts consume the file before it is finished.

In order to avoid this, the files should be produced at a temporary location and when finished it should be moved to the correct location. Moving a file is often an atomic system operation, so it is in general both safe and fast.

Moving files is easiest done with the `renameTo(...)` method in the `java.io.File` class and works as long as the move is in same file system volume.



In other cases the file first has to be copied to the new location and then the original file has to be deleted.

To copy as file with the greatest efficiency, use the `transferTo(...)` and `transferFrom(...)` methods in the `java.nio.channels.FileChannel` class. Deleting files is done with the `delete()` method in the `java.io.File` class.

## 4 Customization

This section describes how to customize the Consistency Checker.

The **Consistency Checker API** is available in `<CheckerHomeDir>/bin/cc_api.jar` and the javadoc for the API is available in `<CheckerHomeDir>/doc/index.html`.

### 4.1 Creating a New Rule

This section describes how to create a new rule. Rules are used in rule based analysis.

#### 4.1.1 Prerequisites

To be able to implement a new rule, access to the Consistency Checker delivery is required. The interfaces that will be used in this guide can be found in the `cc_api.jar` library.

#### 4.1.2 Implementing the Rule

1. Create a class that implements the `com.ericsson.consistency.Rule` interface.
2. (Optional) Implement one or both of the following interfaces in the class:
  - If the rule supports additional parameters during evaluation:  
`com.ericsson.consistency.Parameterizabl`
  - If the rule requires uploading external configurations during evaluation:  
`com.ericsson.consistency.ExternalConfiguration`



**Note:** For information on the interfaces, see **Consistency Checker API** on Page 6.

3. Register the new rule class in the `META-INF/rule.properties` file.
4. Compile and package it into a `.jar` file.

For details about deploying the new rule, see Section 4.1.4 on page 8.

### 4.1.3 Code Examples

The following example uses both Rule and Parameterizable interface.



```
@ExternalFileRequired /* Optional */
public class EqualWithSuffixRule implements Rule, Parameterizable {
    private DefaultAttributeComparator comparator = new DefaultAttributeComparator();
    private String parameter = null;

    @Override
    public boolean evaluate(Object obj1, Object obj2) {
        boolean result = false;
        String leftValue = null;
        String rightValue = null;

        if (obj1 instanceof String && obj2 instanceof String) {
            leftValue = (String) obj1;
            rightValue = (String) obj2;
        } else {
            if (obj1 != null) {
                leftValue = obj1.toString();
            }
            if (obj2 != null) {
                rightValue = obj2.toString();
            }
        }

        if (leftValue == null && rightValue == null) {
            // Then they are equal...
            result = true;
        } else if (leftValue == null) {
            // Then they are not equal...
            result = false;
        } else if (leftValue != null && rightValue != null) {

            Integer i = new Integer(parameter);
            int startIndexLeft = leftValue.length() - i;
            int startIndexRight = rightValue.length() - i;

            String subLeftValue;
            String subRightValue;
            try {
                subLeftValue = leftValue.substring(startIndexLeft, leftValue.length());
                subRightValue = rightValue.substring(startIndexRight, rightValue.length());

                if (comparator.compare(subLeftValue, subRightValue) == 0) {
                    // Then they are equal...
                    result = true;
                } else {
                    // Then they are not equal...
                    result = false;
                }
            } catch (Exception e) {
                result = false;
            }
        } else {
            // Then they are not equal...
            result = false;
        }

        return result;
    }

    @Override
    public void setParameter(String parameter) {
        this.parameter = parameter;
    }

    @Override
    public String getParameter() {
        return parameter;
    }
}
```



#### 4.1.4 Deploying the Rule

The new Rule Class and its helper Classes must be bound in a .jar file and placed in the `<CheckerHomeDir>/bin/lib/` directory.

In order to make the new rule visible in the system make sure it is registered in the META-INF/rule.properties file.

The format in the file:

*<Fully qualified Class name> = <Displayed name in GUI>*

Example:

```
foo.bar.MyRule = My own rule
```

The .jar file containing the rules must be bound into the application .ear files APP\_CC\_Management.ear and APP\_Offline.ear, if Real-time feature is activated it must be bound into APP\_Realtime.ear as well.

1. Create a directory in the `<CheckerHomeDir>/bin` called `lib`.
2. Put the .jar file containing the new rules in the new `<CheckerHomeDir>/bin/lib` directory.

3. Include the .jar file in the Consistency Checker controller application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_CC_Management.ear lib/the-new-rule-lib.jar
```

4. Include the .jar file in the Consistency Checker offline application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_Offline.ear lib/the-new-rule-lib.jar
```

5. Include the .jar file in the Consistency Checker Real-time application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_Realtime.ear lib/the-new-rule-lib.jar
```

6. Redeploy the APP\_CC\_Management.ear, APP\_Offline.ear and APP\_Realtime.ear into the target application server.

## 4.2 Creating a New Data Model

This section describes how to create a new data model. Data models are used in rule based analysis and in Data Extraction.



### 4.2.1 Creating a New Off-line Data Model

A Consistency Checker data model is a Java Bean which describes data in a dump file. The data model class must:

- Implement the `Serializable` interface
- Have a default constructor (without arguments)
- Have getters and setters for each field

In order to create a new Off-line data model, do as follows:

1. Create the Java Bean that corresponds to the target dump file format.
2. Register the new data model in `META-INF/datamodels.properties` file.
3. Compile and package it into a `.jar` file. For details about deploying the new off-line data model, see Section 4.2.3 on page 11.

Code Examples:

```
import java.io.Serializable;
public class MyDataModel implements Serializable {
    private String MSISDN = null;
    private String IMSI = null;
    private String NEW_ATTRIBUTE = null;
    public MyDataModel() {
    }
    public String getMSISDN() {
        return MSISDN;
    }
    public void setMSISDN(String mSISDN) {
        MSISDN = mSISDN;
    }
    public String getIMSI() {
        return IMSI;
    }
    public void setIMSI(String iMSI) {
        IMSI = iMSI;
    }
    public String getNEW_ATTRIBUTE() {
        return NEW_ATTRIBUTE;
    }
    public void setNEW_ATTRIBUTE(String nEW_ATTRIBUTE) {
        NEW_ATTRIBUTE = nEW_ATTRIBUTE;
    }
}
```

### 4.2.2 Creating a New Online Data Model

Creating a new Data model for Online integrations is similar to the one for Off-line. The difference is that Data type classes must be annotated with `@Entity` from the `java.persistence` package.

**Note:** Entity annotated data models can also be used for Off-line extraction and analysis.



To sort multivalue attributes of type String arrays, use the `@OrderBy` annotation.

For details about Deploying the New Online Data Model, see Section 4.2.4 on page 11.

### 4.2.3 Deploying the New Off-line Data Model

To make the new data model visible in the system, register it in the `META-INF/datamodels.properties` file.

The format used in the file is as follows:

*<Fully qualified Class name> = <Displayed name in GUI>*

Example:

```
foo.bar.MyDataModel=My Subscriber
```

Bundle the `.jar` file that contains the new offline data model into the `/lib` directory in the application `.ear` file `APP_Offline.ear`.

Example:

1. Create a directory in the `<CheckerHomeDir>/bin` called `/lib`.
2. Put the `.jar` file containing the new data model in the new `<CheckerHomeDir>/bin/lib` directory.
3. Include the `.jar` file in the Consistency Checker offline application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_Offline.ear lib/the-new-data-model-lib.jar
```

4. Redeploy the `APP_Offline.ear` into the target application server.

**Note:** A data model is often paired with a data parser. This pairing is done by registering the new data model and the corresponding parser to the `META-INF/extractionhandlers.properties` file located in the data parser `.jar` file

For details about data parsers, see Section 4.3 on page 12.

### 4.2.4 Deploying the New Online Data Model

To make the new data model visible in the system, register it in the `META-INF/datamodels.properties` file.

The data model library (`.jar` file) must be included in the `APP_Realtime.ear` file.



Bundle the `.jar` file into the `/lib` directory in `APP_Realtime.ear`.

Example:

1. Create a directory in the `<CheckerHomeDir>/bin` called `/lib`.
2. Put the `.jar` file containing the new data model in the new `<CheckerHomeDir>/bin/lib` directory.
3. Include the `.jar` file in the Consistency Checker Real-time application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_Realtime.ear lib/the-new-data-model-lib.jar
```

4. Redeploy the `APP_Realtime.ear` into the target application server.

Online data source must be defined in the `persistence.xml` file. For more information, see Section 4.5.4 on page 20.

## 4.3 Creating a New Off-line Data Parser

This section describes how to create a new data parser. Data Parsers are used in off-line Data Extraction.

### 4.3.1 Prerequisites

To be able to implement a new data parser, access to the Consistency Checker delivery is required. The class that will be used in this guide can be found in the `cc_api.jar` library.

### 4.3.2 Implementing the New Data Parser

A data parser is aimed for reading from a data source and convert it to the Consistency Checker internal format (default `.csv`).

It is mainly three steps that should be done in order to utilize a new data parser implementation.

1. Create a class that implements the `com.ericsson.consistency.DataParser` interface.
2. Register the new data parser class in the `META-INF/extractionhandlers.properties` file.
3. Compile and package it into a `.jar` file and embed the file in the `APP_Offline.ear/lib` directory.

For available utils, see **Consistency Checker API** on Page 6.





### 4.3.3 Code Examples

A data parser implementation is associated with a data model. In this example we are using the `MyDataModel`, see Section 4.2 on page 9.

In the example the data source is a plain text file.

```
public class MyDataParser implements DataParser {

    private Class<?> dataModelClass = null;
    private String header = "MSISDN          IMSI          NEW_ATTRIBUTE";
    private boolean foundHeader = false;

    @Override
    public void parse(InputStream inStream, ObjectCollector callback)
        throws DataParserException {

        BufferedReader reader = new BufferedReader(new InputStreamReader(inStream));
        String line = null;
        MyDataModel dataModelObject = null;
        try {
            while((line=reader.readLine()) != null) {
                if(foundHeader) {
                    dataModelObject = (MyDataModel)dataModelClass.newInstance();
                    dataModelObject.setMSISDN(line.substring(0, 16).trim());
                    dataModelObject.setIMSI(line.substring(17, 33).trim());
                    dataModelObject.setNEW_ATTRIBUTE(line.substring(34, line.length()).trim());

                    // Write dataModelObject to dumpfile
                    callback.collect((Serializable)dataModelObject);
                }
                else if(line.equals(header)) {
                    foundHeader = true;
                }
            }
        } catch (Exception e) {
            // TODO Error handling
            throw new DataParserException("Error message", e);
        }
    }

    @Override
    public void setDataModel(Class<?> dataModelClass) {
        // NOTE optional to use this, e.g. with multiple data models
        this.dataModelClass = dataModelClass;
    }
}
```

### 4.3.4 Deploying the New Data Parser

New data parsers should be packaged into a `.jar` file and embedded in `APP_Offline.ear/lib` directory.

The new data parser must be registered and associated with a data model. This is done by registering the association to `META-INF/extractionhandlers.properties` entry in the `.jar` file.

The format in the file:

```
<Fully Qualified Class Name for DataModel> = <Fully Qualified
Class Name for DataParser>
```



```
extractionhandlers.properties
#
# <dataModelClass>=<dataParser>
#
foo.bar.example.dm.MyDataModel=foo.bar.example.parser.MyDataParser
```

## 4.4 Generic Extractors

Generic Extractors is a function that extracts data from a database table (SQL) or LDAP entries into Consistency Checker's dump store.

### 4.4.1 Deploy the Generic Extraction Module

The generic extraction module is provided as an optional functionality. In order to be enabled, it needs to be deployed first.

To deploy the generic extraction handlers into consistency checker the `generic_extractors.jar` file should be included in the `APP_Offline.ear` application.

For example:

1. Create a directory called `/lib` in the `<CheckerHomeDir>/bin`.
2. Put the `generic_extractors.jar` to the new `<CheckerHomeDir>/bin/lib` directory.
3. Include the `.jar` file in the Consistency Checker offline application by entering the command from the `<CheckerHomeDir>/bin` directory:

```
jar uf APP_Offline.ear lib/generic_extractors.jar
```

4. Redeploy the `APP_Offline.ear` into the target application server.

### 4.4.2 Place an Extraction Order with the Generic SQL Extraction Handler

One or several JDBC connections must be configured in the target application server in order to use the generic SQL extraction handler. For example, in Glassfish perform the following commands (see Glassfish manual for needed arguments):

```
> asadmin create-jdbc-connection-pool
```

```
> asadmin create-jdbc-resource
```

1. Place an Extraction order and select Generic Extractor as data model (scheduling if needed).
2. Press **Add argument** link and add three additional settings (use the **+** button to add more rows):
  - a Key:



```
com.ericsson.consistency.offline.ExtractionHandler
```

**Value:** com.ericsson.consistency.extract.GenericJDBCE  
xtractor

**b Key:**

```
data_source_jndi
```

**Value:** The JNDI name that used in the `> asadmin create-jdbc-resource` in step 1.

**c Key:**

```
sql_statement
```

**Value:** The SQL query statement that should be used to extract the data.

**3. Submit the order.**

#### 4.4.3

#### Place an Extraction Order with the Generic LDAP Extraction Handler

One or several JNDI resources must be registered and configured in the target application server in order to use the generic LDAP extraction handler. For example, in Glassfish perform the following command (see Glassfish manual for needed arguments):

```
> asadmin create-jndi-resource
```

```
asadmin create-jndi-resource --restype javax.naming.ldap.LdapContext
--factoryclass com.sun.jndi.ldap.LdapCtxFactory
--jndilookupname dc=o,dc=com
--property java.naming.provider.url=ldap\://localhost\://389/:
java.naming.security.authentication=simple:java.naming.security.
principal=cn\=admin,dc\=o,dc\=com:java.naming.security.
credentials=password ldap/myStorageJndi
```

*Example 1 Example of register an External JNDI resource in Glassfish*

1. Place an Extraction order and select Generic Extractor as data model (scheduling if needed).
2. Press **Add argument** link and add four additional settings (use the + button to add more rows):

**a Key:**

```
com.ericsson.consistency.offline.ExtractionHandler
```

**Value:** com.ericsson.consistency.extract.GenericLDAPEx  
tractor



b Key:

`data_source_jndi`

Value: The JNDI name that used in the `> asadmin create-jndi-resource` in step 1.

c Key:

`search_dn`

Value: The DN to the parent of wanted entries.

Example: If all persons that belong to a specific organization should be extracted, the DN shall point to the organization (assuming that the persons are located as child entries in the organization).

d Key:

`search_filter`

Value: Standard search filter specified in RFC 4515. A typical case is to specify the structural object class for person in the example above, that is `(&(objectClass=person))`.

Note that the Generic LDAP Extractor provides support for deep tree aggregation which means that it is possible to specify filter that also includes child entries. Only one entry per level will be aggregated.

e Key:

`identifier_attribute`

Value: The attribute name from the extracted data that will be used as identifier. The extracted data will be indexed by this value and must be unique.

3. Submit the order.

**Note:** Prerequisite for the Generic LDAP Extractor is that the LDAP directory server provides its schema.

## 4.5 Online Data Source Integration

Online Data Sources are used for Real-time data analysis. In order to use this feature, Consistency Checker has to be integrated with the data sources that should be used for Real-time data analysis.

This section and its sub sections describes how to integrate Online data sources for Real-time data analysis.



### 4.5.1 Online Architecture Overview

The least common denominator in Consistency Checker Online support is Java Persistence API (JPA). The JPA contract is used to access data from external sources via Entity annotated Java objects. An Online Data Source is manifested with a JPA Persistent Unit.

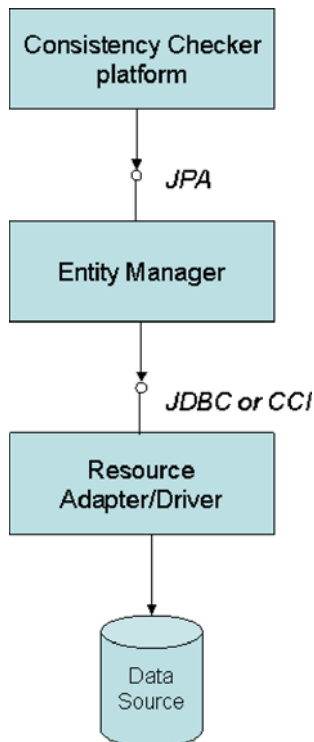


Figure 2 Communication chain in Online data source integrations

Consistency Checker can be integrated with several external data sources at the same time.

In order to avoid compilation required programming when defining new JPA persistence units (or Data sources), Consistency Checker Application managed Entity Managers concept is described in the JPA specification.

### 4.5.2 Integration Scenarios

Though Consistency Checker only depends on JPA for retrieving data from external data sources, it is recommended to follow some standard JEE architectural principles.

#### 4.5.2.1 Integration with a Relation Database

While integrating the Consistency Checker with an ordinary relation database, Java programming is not required. Make sure that the database vendor JDBC



implementation is available in the system and define the configuration in the `persistence.xml` file. For further instructions, see Section 4.5.4 on page 20.

#### **4.5.2.2 Integration with an Enterprise Information System**

The JEE specification uses the term Enterprise Information System (EIS) for arbitrary data sources other than SQL-based. In order to integrate Consistency Checker with EIS data sources, a Resource Adapter compliant with the guide in section Develop Resource Adapter is required.

When the Resource Adapter is deployed, it must be instantiated (normally as pooled connection instances) and associated as resource accessible via JNDI. The Resource Adapter instance must then be associated with Consistency Checker Real-time application via the `persistence.xml` file. For further instructions, see Section 4.5.4 on page 20.

In the Consistency Checker community there are implementations available as “open source” for Ericsson personnel. These implementations can be used as practice and guide development of new Online integrations.

Contributing customer integrations back to the Consistency Checker community increases the Consistency Checker resource portfolio and simplifies further integrations.

#### **4.5.3 Developing Resource Adapter**

A JEE Resource Adapter is a component that is responsible for managing connections to external systems. Consistency Checker Online architecture is prepared for external communication via JEE Resource Adapters based on the JCA 1.5 specification, see Reference [4]. This specification contains optional sections depending on the protocol capabilities, the capabilities of the external system, component reuse ambition, and so on.

In order to have a Resource Adapter compliant to Consistency Checker Online architecture, some optional specifications must be supported.

All aspects of the JCA based Resource Adapters are covered in this document. It is assumed that the integration developer has extended JEE skills in order to develop reliable Resource Adapter.

##### **4.5.3.1 Supported operations**

Currently the Consistency Checker only performs read operations on single data entities. Therefore, if the external system and the used protocol supports other operations, for example, add, delete, and so on, Consistency Checker will not use them.



#### 4.5.3.2 Transaction support

Consistency Checker does not currently use the JEE transaction mechanism. Therefore, the Resource Adapter does not need to support this mechanism. It must not require that transactions are used.

#### 4.5.3.3 Resource Adapter Interface

The specification supports any client interface towards Resource Adapters. However, in order to make use of the Consistency Checker provided common Entity Manager implementation, the Common Client Interface (CCI) named in the JCA 1.5 specification must be used.

#### 4.5.3.4 Security

If the external data source requires authentication, it is recommended not to use the simple `ConnectionFactory` configuration examples via the deployment descriptors, which often can be seen on internet. The reason is that the properties often are stored in plain text in the file system.

It is recommended to use the security mapping concept and getting the credentials via the `java.security.auth.Subject` object.

#### 4.5.3.5 Important highlights in CCI

`ResourceAdapterMetaData.getInteractionSpecsSupported()` - The `InteractionSpec` class at first position (array position 0) will be used.

`Interaction.execute(InteractionSpec ispec, Record input)` - This is the only execute method that will be used by the CCI Entity Manager implementation provided by Consistency Checker. This method must return a `ResultSet`.

`InteractionSpec.setFunctionName(String name)` - It is assumed that the `InteractionSpec` implementation supports the optional attribute `functionName`.

`Record.recordName` - The `recordName` attribute (accessed via `setRecordName` and `getRecordName`) in `Record` implementations carries the data type name (without namespace) or domain information from the external data source, for example, the table name in a relation database or entry name in an LDAP. This means that the Mapped and Indexed Record used as request payload will use `recordName` as holder for the requested data type. The `ResultSet recordName` holds the name of the data type it represents.

`ResultSet` - The `ResultSet` interface contains a large set of methods. In order to minimize the development effort, the method used by the common Entity Manager classes provided by Consistency Checker is listed here:

- `first()`



- `getBoolean(String columnName)`
- `getBytes(String columnName)`
- `getDate(String columnName)`
- `getDouble(String columnName)`
- `getFloat(String columnName)`
- `getInt(String columnName)`
- `getLong(String columnName)`
- `getObject(String columnName)`

Used if the attribute is a multi value like an array or Collection of same primitive.

- `getShort(String columnName)`
- `getString(String columnName)`
- `wasNull()`

#### 4.5.4 Online Data Source Definition

Consistency Checker is using the standard JEE way of defining data sources. That includes:

- Deploy connection adapter/driver
- Associate the Real-time application with pooled connections via JPA's `persistence.xml` configuration.

Consistency Checker uses the Application managed Entity Managers, described in the JPA specification. The Entity Managers configuration in Consistency Checker is fetched from `APP_Realtime.ear/EJB_Realtime.jar/META-INF/persistence.xml`, same as for the Container managed Entity Managers mode. Currently following tags are used:

```
<persistence-unit>
  <provider/>
  <jar-file/>
  <class/>
  <properties>
    <property/>
  </properties>
</persistence-unit>
```

The `persistence.xml` file is normally embedded into the deployable `.ear` file `APP_Realtime.ear/EJB_Realtime.jar/META-INF/persistence.xml`. In order to modify that file, it is extracted from the archive, modified and





inserted into the .ear file. After that the .ear file must be redeployed. In order to avoid all these steps, it is possible to place the persistence.xml into the application server shared classes directory instead. For example, `<Glassfish_Installation>/domains/domain1/lib/classes/META-INF/persistence.xml`

The consequences include the need to restart application server if any changes are done in the persistence.xml file.

#### 4.5.4.1 Defining an SQL Based Data Source

In order to define an SQL based data source, do as follows:

1. Make sure that the data base vendor JDBC driver (adapter) is available on the application server (commercial application servers are often shipped with the most common drivers). If not, see the target application server instructions for deploying JDBC drivers.
2. Create a connection pool in the application server for the target database.
3. Update the persistence.xml located in the APP\_Realtime.ear/EJB\_Realtime.jar/META-INF/persistence.xml

Consistency Checker will automatically scan the system for information in the persistence.xml file in order to populate the MBean register of DataSources.

**Note:** It is important that data sources and their supported data types are registered in the Consistency Checker data source registry, otherwise the Online feature will not work.

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">

  <persistence-unit name="My_DataSource 1" transaction-type="RESOURCE_LOCAL">
    <provider>oracle.toplink.essentials.ejb.cmp3.EntityManagerFactoryProvider</provider>

    <jar-file>lib/MyDataModel.jar</jar-file>

    <properties>
      <property name="toplink.jdbc.user" value="league"/>
      <property name="toplink.jdbc.password" value="league"/>
      <property name="toplink.jdbc.url" value="jdbc:derby://localhost:1527/league"/>

      <property name="toplink.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="toplink.ddl-generation" value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

#### *Example 2 Using the JDBC based data source in Consistency Checker*

The possible properties and their values depend on the persistence provider implementation and in some cases on the JDBC implementation. For more information, see the chosen persistence provider documentation.



#### 4.5.4.2 Defining a Resource Adapter Based Data Source

This section describes the procedure when Resource Adapters developed according to instructions in this document are used.

In order to a Resource Adapter based data source, do as follows:

1. Deploy the Resource Adapter into the application server according to the target application server deploy instructions.
2. Create a connection pool in the application server for the target data source.
3. Update the `persistence.xml` located in `APP_Realtime.ear/EJB_Realtime.jar/META-INF/persistence.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/
  persistence/persistence_1_0.xsd"
  version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">

  <persistence-unit name="My_DataSource 1" transaction-type="RESOURCE_LOCAL">
    <provider>foo.bar.PersistenceProviderImpl</provider>

    <jar-file>lib/MyDataModel1.jar</jar-file>
    <jar-file>lib/MyDataModel2.jar</jar-file>

    <properties>
      <property name="ericsson.cci.resource_jndi" value="MyDSConResource"/>
      <property name="ericsson.cci.select_function" value="GET"/>
    </properties>

  </persistence-unit>
</persistence>
```

##### *Example 3 Using Data Source accessed via the Resource Adapter*

The CCI Entity Manager implementation provided in Consistency Checker requires following properties:

- `ericsson.cci.resource_jndi` – The JNDI name to the Resource Adapter ConnectionFactory.
- `ericsson.cci.select_function` – The function name the Resource Adapter will use in its `InteractionSpecs` implementation.

## 4.6 Notification

Consistency Checker provides support for notification events via JMS. The support can be used for post-processing of analysis results. In order to receive notification events, a JMS message consumer has to be implemented. The easiest way to implement a JMS message consumer is to implement an Message Driven Bean (MDB). Notifications events are available via the JMS Topic Destination Resource `jms/consistency/event`.

The use of a JMS Topic enables multicast notifications, thus, there can be multiple event consumers independently subscribing on the same notifications.



There are five different types of notification events:

- `com.ericsson.consistency.event.StartOrderEvent` - contains the order object and the absolute path to the current result file.
- `com.ericsson.consistency.event.InterruptedEvent` - contains `orderId`.
- `com.ericsson.consistency.event.OrderFinishedEvent` - contains the report object.
- `com.ericsson.consistency.event.ArtifactCreatedEvent` - contains artifacts that are created in the system, for example, when a new order is submitted to the system.
- `com.ericsson.consistency.event.ArtifactDeletedEvent` - contains artifacts that are deleted from the system, for example, when an order is removed from the system.

Example code of an notification event consumer MDB:

```
@MessageDriven(activationConfig = { @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Topic" ), mappedName = "jms/consistency/event" })
public class EventConsumer implements MessageListener {
    private static final Logger log=Logger.getLogger(EventConsumer.class.getName());

    public void onMessage(Message message) {
        ObjectMessage objMsg=(ObjectMessage)message;
        try {
            log.info(getClass().getSimpleName() + " got an event message of type " +objMsg.getObject().getClass().getName());
            if (objMsg.getObject() instanceof OrderFinishedEvent) {
                OrderFinishedEvent event = (OrderFinishedEvent) objMsg.getObject();
                log.info("Order finished: " + event.getReport().getOrder().getIdentifier() );

                //TODO Here is the place for post processing code.
            }
            else if (objMsg.getObject() instanceof StartOrderEvent) {
                StartOrderEvent startEvent = (StartOrderEvent) objMsg.getObject();
                log.info("Inconsistency results will be written to: " + startEvent.getResultFilePath() );

                //TODO Here is the place for code that needs to know when processing an order is started.
            }
            //etc...

        } catch (JMSException e) {
            log.log(Level.SEVERE, "Failed to receive message.",e);
        }
    }
}
```

Notification events can also be filtered with JMS message tags. Each notification message is tagged with the following String property:

```
msg.setStringProperty("EVENT_TYPE",event.getClass().getName());
```

Message filtering on the MDB is enabled with the following annotation:



```
@MessageDriven(activationConfig = { @ActivationConfigProperty(propertyName =
"destinationType", propertyValue = "javax.jms.Queue"),
@ActivationConfigProperty(propertyName = "messageSelector",
propertyValue = "EVENT_TYPE = 'com.ericsson.consistency.event.StartOrderEvent'") },
mappedName = "jms/consistency/event")
```

With this annotation, the MDB will only accept messages with the String property `EVENT_TYPE = 'com.ericsson.consistency.event.StartOrderEvent'`.

## 4.7 GUI Language

Consistency Checker GUI have multiple language support. In order to add a language follow instructions below:

1. Unpack application:
  - a. Open/unpack the `APP_CC_Managment.ear` file.
  - b. Open/unpack the `web_gui.war` file embedded in `APP_CC_Managment.ear` to a local directory.
2. Translate text:
  - a. Go to the `locales` folder of the extracted directory.
  - b. Create a copy of the `en-us` folder and rename it to a desired language code (2 letters) abbreviation, for example, `sv` for Swedish.
  - c. Open the folder created for the new language. This folder contains language files (`.json`) in it and its sub-folders.  
  
Each `.json` language file represents a part of the Consistency Checker GUI, and contains a bunch of pairs as follow:  
`"<GUI_object_name>": "<English translation>"`
  - d. For each `.json` language file, open it in a text editor, and change all the `"<English translation>"` to the desired language translation.

### Note:

- The `"<GUI_object_name>"` parts must be kept as the same.
- Use Unicode code point when translating to characters that are not supported for UTF-8. For example, use `\u00F1` to represent ñ.

Example of the `container.json` file before and after translation.

### Before



```
{
  "confirmNavigation": "Confirm Navigation",
  "confirmNavigationMessage": "Are you sure that you wan",
  "leaveThisPage": "Leave this Page",
  "stayOnThisPage": "Stay on this Page"
}
```

#### After

```
{
  "confirmNavigation": "Bekräfta Navigation",
  "confirmNavigationMessage": "Är du säker på att du vil",
  "leaveThisPage": "Lämna denna sida",
  "stayOnThisPage": "Stanna på denna sida"
}
```

### 3. Enable the added language:

- a. Go to the extracted directory, open the configuration file `web_gui\config.js` in a text editor.
- b. Search for `locales` keys, and add the new language code in their values. For example:

```
'locales': ['en-us', 'sv']
```

**Note:** There are two places to change.

4. Repack the `web_gui.war` and `APP_CC_Management.ear`.
5. Redeploy the `APP_CC_Managment.ear` in the application server, refer to the Glassfish documentation.

## 4.8 Excel Reports

Consistency Checker supports generating human readable analysis reports in customizable Microsoft Excel compliant format. This feature is enabled by default.

When this feature is enabled, an Excel (.xlsx) workbook is exported as a post processing task for Analysis Orders and will be started directly after the processing of an Analysis Order. The exported report presents the same information that is shown in the GUI after performing the analysis.

**Note:** Large result files may take several minutes to export.

The output workbook may be customized to follow company branding with styling and positioning.



### 4.8.1 Disable the Export of Excel Reports

To disable the export of Excel reports, in the `<CheckerHomeDir>/conf/ccengine.properties`, uncomment and change the property value to `false`.

**Example:**

```
com.ericsson.consistency.export.excel=false
```

**Note:** To apply the change, restart the application server.

### 4.8.2 Customize Excel Reports

This section describes how to customize the Excel report template.

The Excel format is customizable through a template containing anchor tags. The template, `template.xlsx`, is located in the `<CheckerHomeDir>/conf/` directory. The template must be in the `.xlsx` format defined in the Office Open XML (OOXML) specification, see <http://www.ecma-international.org/publications/standards/Ecma-376.htm>, Reference [5]. The `.xlsx` format is supported from MS Excel 2007 (v12.0).

To customize the template, update or replace the existing template by inserting and styling the anchor tags as desired. Where the anchor tags are placed, data will be inserted. It is possible to use any sub-set of the anchor tags.

The supporter anchor tags are shown in Table 3.

*Table 3 Supported Anchor Tags*

Anchor Tag	Description
<ORDER_ID>	Order info - The order identifier.
<ORDER_CREATION>	Order info - The date and time when the order was created.
<DESCRIPTION>	Order info - The description text from the order.
<SCHEDUING>	Order info - The scheduling type (ONCE, NOW, DAILY, WEEKLY, MONTHLY).
<START_TIME>	Report info - The start time of the analysis job.
<END_TIME>	Report info - The finished time of the analysis job.
<STATUS>	Report info - The status of the analysis job (SUCCESS, FAILED).



Anchor Tag	Description
<SOURCE_A>	Report info - The data source A.  In Offline this is the absolute path to file in dump store.  In Real time this is the resource id from <code>persistence.xml</code> .
<SOURCE_B>	Report info - The data source B.  In Offline this is the absolute path to file in dump store.  In Real time this is the resource id from <code>persistence.xml</code> .
<POSTS_IN_A>	Report info - Number of read posts from data source A.
<POSTS_IN_B>	Report info - Number of read posts from data source B.
<SUM_COMPARABLE>	Report info - List of posts that are comparable (posts that have same identifier in both data sources).
<SUM_UNIQUE_IN_A>	Report info - List of posts that are unique in data source A.
<SUM_UNIQUE_IN_B>	Report info - List of posts that are unique in data source B.
<SUM_MISMATCHED>	Result info - Number of posts that are comparable but deviates summarized by rule or pattern.
<LIST_DEVIATION>	Report info - List of attributes that are comparable but deviates from the rules or pattern. <sup>(1)</sup>
<LIST_UNIQUE_IN_A>	Result info - List of all unique identifiers from data source A.  For further information, see Page 28.
<LIST_UNIQUE_IN_B>	Result info - List of all unique identifiers from data source B.  For further information, see Page 28.
<LIST_MISMATCH>	Result info - List of all attribute deviations.  For further information, see Page 28.

(1) New rows will be inserted at this point. Do not add additional information in other cells on the same row.



**Note:** Anchor tags `<LIST_UNIQUE_IN_A>`, `<LIST_UNIQUE_IN_B>` and `<LIST_MISMATCH>` are only supported if report format 2 is used (default). Since Realtime Analysis does not support this format, these tags will be ignored in the Realtime Analysis reports. Sheets where this anchor tag is inserted will be replaced and is not possible to customize. No other anchor tags can be used on the same sheet.

## 5 Definitions

This section describes the definitions of the files produced by the Consistency Checker.

### 5.1 Dump File Format

Dump files can be manually verified with the `<CheckerHomeDir>/bin/analyzecsv` tool.

The default dump store is `<CheckerHomeDir>/var/dump`. The files placed in this directory has the platform default character encoding.

The default dump files have the **comma-separated values** (CSV) format, where the value separator is a comma character. . If the file contains multi-value attributes the `|` (pipe) character is used for separation, for example, `A, B, C1 | C2 | C3, D, E`.

The first row is comprised of delimited fields separated by the comma character. Each row is separated by a new line.

Values that must be escaped during the extraction process are: comma character, carriage return and new lines.

The CSV file can contain a header row, but the header row must then comply to the following rules:

- Names can contain letters, numbers, and other characters.
- Names cannot start with a number or punctuation character.
- Names cannot start with the letters `xml` (or `XML`, `Xml`, and so on).
- Names cannot contain spaces.

Empty columns are considered as null values.





### Example:

```
sub_MSISDN,Acct_MSISDN,IMSI,state,sub_status,language,passCode
511111111,624121131,642024511111111,ADDITIONAL,1,1,pincode1
511111112,624121132,,NOT_CONNECTED,0,2,pincode1
511111113,624121133,642024511111113,CONNECTED,1,0,pincode1
511111114,624121134,642024511111114,CONNECTED,0,1,pincode1
511111115,624121135,642024511111115,CONNECTED,1,2,pincode1
...
```

## 5.2 Rule Based Analysis

### 5.2.1 Rule Based Analysis Result File Format

The result is represented by an compressed (.zip) XML file.

The files are located in the `<CheckerHomeDir>/var/reports` directory.

The file suffix is `_Result.zip`, see also *System Administrators Guide for Consistency Checker*, Reference [1].

The information in the file is represented in the Header, Footer and CC elements.

### Example:

```
<Header>
<Specification>SeleniumDataComparisonSpecification</Specification>
<OrderId>SeleniumCheckComparison</OrderId>
<ComparedFiles>
  <ComparedFiles id="A0">var/dump/SeleniumExtractComparison_HLR1.log_2010-11-10_12.57.24.csv
</ComparedFiles>
  <ComparedFiles id="A1">var/dump/SeleniumExtractComparison_SDPDUMP_2.zip_2010-11-10_12.57.24.csv
</ComparedFiles>
</Header>
<CC key="111665596" uniqueentity="true">
  <dump ref="A1">
    <Language>1</Language>
    <Subscriber_MSISDN>111665596</Subscriber_MSISDN>
    <Account_MSISDN>111665596</Account_MSISDN>
    <Subscriber_status>0</Subscriber_status>
    <Refill_failed>0</Refill_failed>
    <Refill_bar_end>0</Refill_bar_end>
    <First_ivr_call_done>0</First_ivr_call_done>
    <First_call_done>0</First_call_done>
    <Special_announc_played>0</Special_announc_played>
    <Sfee_warn_played>0</Sfee_warn_played>
    <Sup_warn_played>0</Sup_warn_played>
    <Low_level_warn_played>0</Low_level_warn_played>
    <Wanted_block_status>30</Wanted_block_status>
    <Actual_block_status>30</Actual_block_status>
    <Eoc_selection_id>255</Eoc_selection_id>
    <Pin_code />
  </dump>
</CC>
<CC key="111665597" uniqueentity="true">
  <dump ref="A1">
    <Language>1</Language>
    <Subscriber_MSISDN>111665597</Subscriber_MSISDN>
    <Account_MSISDN>111665597</Account_MSISDN>
    <Subscriber_status>0</Subscriber_status>
    <Refill_failed>0</Refill_failed>
    <Refill_bar_end>0</Refill_bar_end>
    <First_ivr_call_done>0</First_ivr_call_done>
```



```
<First_call_done>0</First_call_done>
<Special_announc_played>0</Special_announc_played>
<Sfee_warn_played>0</Sfee_warn_played>
<Sup_warn_played>0</Sup_warn_played>
<Low_level_warn_played>0</Low_level_warn_played>
<Wanted_block_status>30</Wanted_block_status>
<Actual_block_status>30</Actual_block_status>
<Eoc_selection_id>255</Eoc_selection_id>
<Pin_code />
</dump>
</CC>
<CC key="111665598" uniqueentity="true">
<dump ref="A1">
  <Language>1</Language>
  <Subscriber_MSISDN>111665598</Subscriber_MSISDN>
  <Account_MSISDN>111665598</Account_MSISDN>
  <Subscriber_status>0</Subscriber_status>
  <Refill_failed>0</Refill_failed>
  <Refill_bar_end>0</Refill_bar_end>
  <First_ivr_call_done>0</First_ivr_call_done>
  <First_call_done>0</First_call_done>
  <Special_announc_played>0</Special_announc_played>
  <Sfee_warn_played>0</Sfee_warn_played>
  <Sup_warn_played>0</Sup_warn_played>
  <Low_level_warn_played>0</Low_level_warn_played>
  <Wanted_block_status>30</Wanted_block_status>
  <Actual_block_status>30</Actual_block_status>
  <Eoc_selection_id>255</Eoc_selection_id>
  <Pin_code />
</dump>
</CC>
<CC key="111665599" uniqueentity="true">
<dump ref="A1">
  <Language>1</Language>
  <Subscriber_MSISDN>111665599</Subscriber_MSISDN>
  <Account_MSISDN>111665599</Account_MSISDN>
  <Subscriber_status>0</Subscriber_status>
  <Refill_failed>4</Refill_failed>
  <Refill_bar_end>0</Refill_bar_end>
  <First_ivr_call_done>0</First_ivr_call_done>
  <First_call_done>0</First_call_done>
  <Special_announc_played>0</Special_announc_played>
  <Sfee_warn_played>0</Sfee_warn_played>
  <Sup_warn_played>0</Sup_warn_played>
  <Low_level_warn_played>0</Low_level_warn_played>
  <Wanted_block_status>7936</Wanted_block_status>
  <Actual_block_status>22</Actual_block_status>
  <Eoc_selection_id>255</Eoc_selection_id>
  <Pin_code />
</dump>
</CC>
<CC key="17709080000" uniqueentity="true">
<dump ref="A1">
  <Language>1</Language>
  <Subscriber_MSISDN>17709080000</Subscriber_MSISDN>
  <Account_MSISDN>17709080000</Account_MSISDN>
  <Subscriber_status>0</Subscriber_status>
  <Refill_failed>0</Refill_failed>
  <Refill_bar_end>0</Refill_bar_end>
  <First_ivr_call_done>0</First_ivr_call_done>
  <First_call_done>0</First_call_done>
  <Special_announc_played>0</Special_announc_played>
  <Sfee_warn_played>0</Sfee_warn_played>
  <Sup_warn_played>0</Sup_warn_played>
  <Low_level_warn_played>0</Low_level_warn_played>
  <Wanted_block_status>28</Wanted_block_status>
  <Actual_block_status>28</Actual_block_status>
  <Eoc_selection_id>100</Eoc_selection_id>
  <Pin_code />
</dump>
</CC>
<CC key="30115599" uniqueentity="true">
<dump ref="A1">
  <Language>1</Language>
  <Subscriber_MSISDN>30115599</Subscriber_MSISDN>
  <Account_MSISDN>30115599</Account_MSISDN>
```



```

<Subscriber_status>0</Subscriber_status>
<Refill_failed>0</Refill_failed>
<Refill_bar_end>0</Refill_bar_end>
<First_ivr_call_done>0</First_ivr_call_done>
<First_call_done>0</First_call_done>
<Special_announc_played>0</Special_announc_played>
<Sfee_warn_played>0</Sfee_warn_played>
<Sup_warn_played>0</Sup_warn_played>
<Low_level_warn_played>0</Low_level_warn_played>
<Wanted_block_status>30</Wanted_block_status>
<Actual_block_status>30</Actual_block_status>
<Eoc_selection_id>255</Eoc_selection_id>
<Pin_code />
</dump>
</CC>
<CC key="511111111" uniqueentity="true">
  <dump ref="A0">
    <MSISDN>511111111</MSISDN>
    <IMSI>642024511111111</IMSI>
    <STATE>ADDITIONAL</STATE>
    <AUTHD>AVAILABLE</AUTHD>
  </dump>
</CC>
<CC key="511111112" uniqueentity="true">
  <dump ref="A0">
    <MSISDN>511111112</MSISDN>
    <IMSI />
    <STATE>NOT CONNECTED</STATE>
    <AUTHD />
  </dump>
</CC>
<CC key="511111113" uniqueentity="true">
  <dump ref="A0">
    <MSISDN>511111113</MSISDN>
    <IMSI>642024511111113</IMSI>
    <STATE>CONNECTED</STATE>
    <AUTHD>AVAILABLE</AUTHD>
  </dump>
</CC>
<CC key="511111114" uniqueentity="true">
  <dump ref="A0">
    <MSISDN>511111114</MSISDN>
    <IMSI>642024511111114</IMSI>
    <STATE>CONNECTED</STATE>
    <AUTHD>AVAILABLE</AUTHD>
  </dump>
</CC>
<CC key="511111115" uniqueentity="true">
  <dump ref="A0">
    <MSISDN>511111115</MSISDN>
    <IMSI>642024511111115</IMSI>
    <STATE>CONNECTED</STATE>
    <AUTHD>NO ACCESS TO AUC</AUTHD>
  </dump>
</CC>
<Footer>
  <EndTime>2010-11-10 12:57:57</EndTime>
  <StartTime>2010-11-10 12:57:56</StartTime>
</Footer>
</raw_result>

```

### 5.2.1.1 Rule Based Analysis Result Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="raw_result">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Header" />
        <xs:element maxOccurs="unbounded" ref="CC" />
        <xs:element ref="Footer" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```



```
</xs:element>
<xs:annotation>
  <xs:documentation>
    The Header information is always present.
  </xs:documentation>
</xs:annotation>
<xs:element name="Header">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Specification" />
      <xs:element ref="OrderId" />
      <xs:element ref="ComparedFiles" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    The Consistency Check Specification used for this order.
  </xs:documentation>
</xs:annotation>
<xs:element name="Specification" type="xs:NCName" />
<xs:annotation>
  <xs:documentation>
    The identifier of the check order job.
  </xs:documentation>
</xs:annotation>
<xs:element name="OrderId" type="xs:NCName" />
<xs:annotation>
  <xs:documentation>
    The CC information is only present if an entity is unique (uniqueentity=true) or if two
    entities from two dump files has an attribute mismatch (entitiesmismatch=true).
    The key is the identifier of the entity. Specific dynamic information will also be
    present.
  </xs:documentation>
</xs:annotation>
<xs:element name="CC">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="dump" />
    </xs:sequence>
    <xs:attribute name="entitiesmismatch" type="xs:boolean" />
    <xs:attribute name="key" use="required" type="xs:string" />
    <xs:attribute name="uniqueentity" type="xs:boolean" />
    <xs:attribute name="type" >
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="FAILED"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    The ref attribute in the dump element points to the id attribute defined in ComparedFiles.
  </xs:documentation>
</xs:annotation>
<xs:element name="dump">
  <xs:complexType>
    <xs:sequence>
      <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="ref" use="required" type="xs:NCName" />
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    The Footer information is always present.
  </xs:documentation>
</xs:annotation>
<xs:element name="Footer">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="EndTime" />
      <xs:element ref="StartTime" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```

</xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    The end time of the check order job with format: YYYY-MM-DD hh:mm:ss.
  </xs:documentation>
</xs:annotation>
<xs:element name="EndTime" type="xs:string" />
<xs:annotation>
  <xs:documentation>
    The start time of the check order job with format: YYYY-MM-DD hh:mm:ss.
  </xs:documentation>
</xs:annotation>
<xs:element name="StartTime" type="xs:string" />
<xs:annotation>
  <xs:documentation>
    The dump files used during the Consistency check job.
    The attribute id represent each of the dump files.
  </xs:documentation>
</xs:annotation>
<xs:element name="ComparedFiles">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="ComparedFiles" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:NCName" />
  </xs:complexType>
</xs:element>
</xs:schema>

```

## 5.3 Pattern Based Analysis

### 5.3.1 Pattern Based Analysis Result File Format

The result is represented by an compressed (.zip) XML file.

The files are located in the `<CheckerHomeDir>/var/reports` directory.

The file suffix is `_Result.zip`, see also *System Administrators Guide for Consistency Checker*, Reference [1].

Example:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<raw_result>
  <Header>
    <OrderId>Example1</OrderId>
    <ComparedFiles>
      <ComparedFiles id="A0">C:\prog\ConsistencyChecker\var\dump\dsA.csv
    </ComparedFiles>
      <ComparedFiles id="A1">C:\prog\ConsistencyChecker\var\dump\dsB.csv
    </ComparedFiles>
    </ComparedFiles>
  </Header>
  <CC key="1011111112" uniqueentity="true">
    <dump ref="A0">
      <IMSI>9900001</IMSI>
      <Surname>larry</Surname>
      <FamilyName>MILFORD</FamilyName>
      <Street>Wisteria</Street>
      <ZIPcode>55335</ZIPcode>
      <City>GIBBON</City>
      <PhoneNumber>0799999999</PhoneNumber>
      <PrivateID>1011111112</PrivateID>
      <State>1</State>
    </dump>
  </CC>
</raw_result>

```



```
<MISMATCH key="1111112111">
  <HEAD A_ref="A0" B_ref="A1">
    <B_ATTR name="PrivateID">1111112111</B_ATTR>
    <B_ATTR name="PublicID">+46799998999</B_ATTR>
    <B_ATTR name="SIPURI">111112111@domain.com</B_ATTR>
    <B_ATTR name="Mail">fernando.marciano@domain.com</B_ATTR>
    <B_ATTR name="State">0</B_ATTR>
  </HEAD>
  <A_ATTR name="IMSI" value="10001000">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="Surname" value="FERNANDO">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="FamilyName" value="MARCIANO">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="Street" value="San Carlos Drive">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="ZIPcode" value="41645">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="City" value="LANGLEY">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="PhoneNumber" value="0799998998">
    <X />
    <X status="NOK" />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="PrivateID" value="1111112111">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
  <A_ATTR name="State" value="0">
    <X />
    <X />
    <X />
    <X />
    <X />
  </A_ATTR>
</MISMATCH>
.
.
.
```



```

    <Footer>
      <StartTime>2013-05-15 11:10:51</StartTime>
      <EndTime>2013-05-15 11:11:06</EndTime>
    </Footer>
  </raw_result>

```

### 5.3.1.1 Pattern Based Analysis Result Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="raw_result" type="raw_resultType"/>

  <xs:complexType name="raw_resultType">
    <xs:choice maxOccurs="unbounded" minOccurs="0">
      <xs:element type="HeaderType" name="Header"/>
      <xs:element type="CCType" name="CC" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element type="MISMATCHType" name="MISMATCH" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element type="FooterType" name="Footer"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="HeaderType">
    <xs:sequence>
      <xs:element type="xs:string" name="OrderId"/>
      <xs:element type="ComparedFilesType" name="ComparedFiles"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="CCType">
    <xs:sequence>
      <xs:element type="HEADType" name="HEAD"/>
      <xs:element type="A_ATTRType" name="A_ATTR" maxOccurs="unbounded" minOccurs="1"/>
      <xs:element type="dumpType" name="dump" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:int" name="key" use="optional"/>
    <xs:attribute type="xs:string" name="uniqueentity" use="optional"/>
  </xs:complexType>

  <xs:complexType name="ComparedFilesType" mixed="true">
    <xs:sequence>
      <xs:element type="ComparedFilesType" name="ComparedFiles" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="id"/>
  </xs:complexType>

  <xs:complexType name="dumpType">
    <xs:sequence>
      <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute type="xs:string" name="ref"/>
  </xs:complexType>

  <xs:complexType name="B_ATTRType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="name"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="HEADType">
    <xs:sequence>
      <xs:element type="B_ATTRType" name="B_ATTR" maxOccurs="unbounded" minOccurs="1"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="A_ref" />
    <xs:attribute type="xs:string" name="B_ref" />
  </xs:complexType>

```



```
</xs:complexType>

<xs:complexType name="A_ATTRType">
  <xs:sequence>
    <xs:element name="X" maxOccurs="unbounded" minOccurs="1">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute type="xs:string" name="status" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute type="xs:string" name="name"/>
  <xs:attribute type="xs:string" name="value" use="optional"/>
</xs:complexType>

<xs:complexType name="XType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute type="xs:string" name="status" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="MISMATCHType">
  <xs:sequence>
    <xs:element type="HEADType" name="HEAD" minOccurs="0"/>
    <xs:element type="A_ATTRType" name="A_ATTR" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element type="dumpType" name="dump" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute type="xs:int" name="key"/>
  <xs:attribute type="xs:string" name="uniqueentity" use="optional"/>
</xs:complexType>

<xs:complexType name="FooterType">
  <xs:sequence>
    <xs:element type="xs:string" name="StartTime"/>
    <xs:element type="xs:string" name="EndTime"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```





## Reference List

### Ericsson Documents

- [1] *System Administrators Guide for Consistency Checker*, 5/1543-CSH 109 628 Uen
- [2] *Function Specification Consistency Checker*, 21/155 17-CSH 109 628 Uen

### 3PP Documents

- [3] *Encoded Java Beans*, <http://docs.oracle.com/javase/6/docs/api/java/beans/XMLEncoder.html/>
- [4] *JEE Connector Architecture*, [http://download.oracle.com/otndocs/jcp/connector\\_architecture-1.6-fr-oth-JSpec/](http://download.oracle.com/otndocs/jcp/connector_architecture-1.6-fr-oth-JSpec/)
- [5] *Office Open XML (OOXML) Specification*, <http://www.ecma-international.org/publications/standards/Ecma-376.htm>