i-net Clear Reports 2015

Programming Guide

Server

i-net software

1 Overview 2
1.1 Sample files 2
1.2 Requirements 2
1.3 Call Sequence 2
2 Servlet Filter 4
3 Cache 5
4 EngineFactory 7
5 PropertiesChecker 8
5.1 Send an e-mail in case of an error 8
6 DataFactory 10
6.1 Setting the DataFactory class 10
6.1.1 Data Source Configuration 10
6.1.2 API 10
6.2 Implementation of your own DataFactory 11
7 Dummy JDBC Driver 12
7.1 Building your own JDBC driver 12
8 EngineRenderData 13
9 CacheRenderData 14
10 RDC and Engine 15
10.1 Units in RDC 15
10.1.1 Twips, Inches and Millimeters 15
10.1.2 Conversion 15
11 Compare the programming techniques 16
12 Formula Expander Class(es) - Formula Functions 17
12.1 Examples: 18
13 Plugins 20
13.1 Architecture 20
13.2 Live cycle 20
13.3 Plugin Properties 21
13.4 Extension points 22
13.5 Sample Code 23

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

1 Overview

I-net Clear Reports contains an extensive API for programming the report server. This guide will give you an overview about the possibilities. More complete information can be found in the documentation, e.g. the list of libraries and the API documentation. You can find the documentation in the installation directory (if installed) or on the i-net Clear Reports website.

1.1 Sample files

Please find the examples of this guide as java classes and some more examples in the folder "/documentation/developer/samples" of the installation folder. (If you have installed the Documentation and Examples)

1.2 Requirements

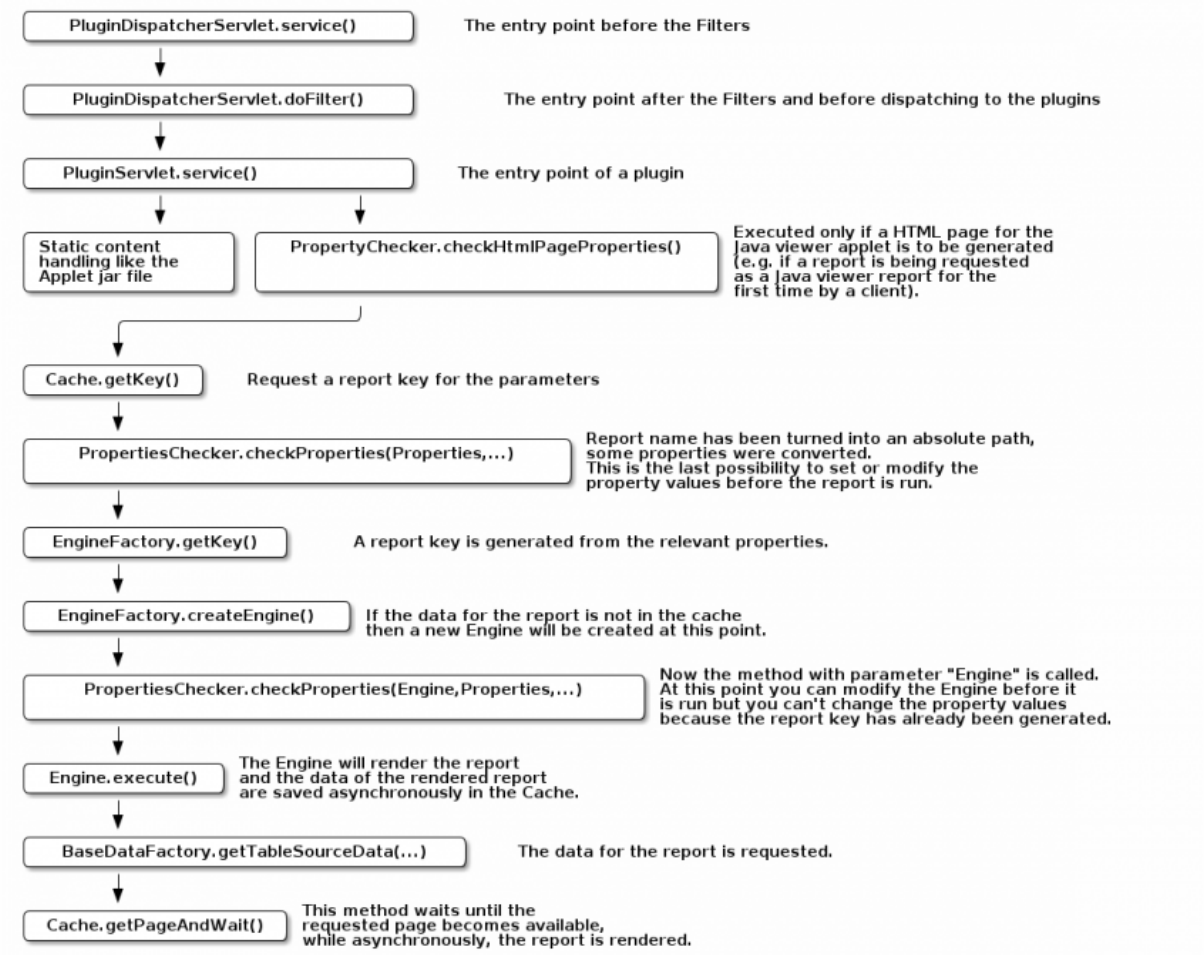
This guide describes the usage of the i-net Clear Reports API. Therefore it is necessary that i-net Clear Reports is installed either as report server or it is embedded into your own application so that you can access the API.

Note that the main JAR file to link your source to is ClearReports.jar and inetcore.jar, found in the core subdirectory of your installation. But you need more jar files from the core directory if you want embedded it in your application. For details see at Libraries of i-net Clear Reports.

1.3 Call Sequence

To help you understand where you can change what, we'll show you first the sequence of code points of a typical report request. Many of these methods (as for example the PropertyChecker methods) can be overloaded by your own classes.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.



© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

2 Servlet Filter

Servlet filters are executed before all other classes. The servlet filters has an effect both on the rendering as well as on the remote access.

Any implementation of the class javax.servlet.Filter can be used as servlet filter. There are a lot of third party implementations available.

To add a servlet filter to i-net Clear Reports, you should write a plugin and register your filter in the ServerPluginManager.

Alternatively put a .jar file with the filter into the folder: "<install-dir>/lib". In the dialog "Customization" it is possible to specify a comma separated list of servlet filters. Add the class name of the filter to the property "Servlet Filter".

Using a servlet filter it is possible to forward a successful log-in to a front-end server to i-net Clear Reports so that a temporary data source can be created for the logged in user (session scope datasource).
Using a servlet filter you have complete control over the input(javax.servlet.HttpServletRequest) and output(javax.servlet.HttpServletResponse).

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

3 Cache

For some output formats like the Applet Viewer or HTML the server will send the rendering result in multiple steps. Therefore it is necessary to cache the results between different request. Also, with the cache it is possible to share the rendered results between multiple users. This optional feature can reduce the load on the server and it can be enabled/disabled using the Configuration property: Reload On New Request.

You can find the cache API in the API documentation.

The following sample shows a simple usage of the Cache API to export to PDF:

```
Cache cache = Cache.getCache();
Properties reportProperties = new Properties();
reportProperties.put("report", "c:\\myreport.rpt" );
reportProperties.put("export_fmt", "pdf" );
ReportCacheKey key = cache.getKey(reportProperties);

int pageCount = cache.getPageCountAndWait(key);
for (int i = 1; i <= pageCount; i++) {
    byte[] page = cache.getPageAndWait(key, i);
    ....
}
```

More samples for the usage of the Cache API can be found in the Java code samples directory: "<installation directory>/client/documentation/samples/cache".

If you use the Cache API directly you don't need to implement the caching yourself. You'll have full control over the connection between client and server.

You can:

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

- set or modify the report name
- set or modify the report parameters (prompts)
- modify the HTML page of the applet
- create the Engine (limited with addEngine)

You can not:

- use the Remote Configuration Manager

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

4 EngineFactory

The Cache API has 2 static methods getEngineFactory and setEngineFactory. With these methods you can register your own global EngineFactory. You can find the EngineFactory API in the API documentation. It is recommended to extend com.inet.report.cache.EngineFactoryImpl instead of implementing an EngineFactory from scratch.

With the EngineFactory you can:

- use your own report templates
- use your own parameters
- use the Remote Configuration Manager
- control the Engine creation
- skip some parameters from the report key

You can not:

- modify the HTML page of the applet

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

5 PropertiesChecker

Another option is to implement the PropertiesChecker interface.

You can find the PropertiesChecker API in the API documentation.

There are multiple solutions to set a custom PropertyChecker:

- set the class name in the property Property Checker of the currently used Configuration

With the PropertyChecker you can accomplish much the same as with an overridden ReportServlet, though ReportServlet has more interaction points.

You can:

- set or modify the report name
- set or modify the report parameters (prompts)
- modify the HTML page of the applet
- modify the Engine after loading
- force a specific output format (applet, pdf, etc.)
- use the Remote Configuration Manager
- register an EngineFinishListener using Engine.addFinishListener(EngineFinishListener)

You can not:

- create the Engine

5.1 Send an e-mail in case of an error

In the method checkProperties(Engine, Properties, Object) of the PropertyChecker you can register an EngineFinishListener to the current engine (Engine.addFinishListener(EngineFinishListener)).

In the method engineFinish(EngineFinishEvent event) of this

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

EngineFinishListener you can use the method Engine.getErrorMsg() to get the error message, if an error has occurred. If no error has occurred, this method returns a Null.

In case of an error you can use this error message in your email.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

6 DataFactory

If you want to manually set the data for a report then you can do it with any of the Engine.setData() methods. An alternative method is to set a data factory.

An instance of the class com.inet.report.DataFactory is the factory for the report data. Most implementations of the DataFactory use JDBC but it is also possible to set the data without JDBC. The advantages of the DataFactory class is that it is easier to set the data for sub reports. This is the only solution for setting individual data for each sub report.

6.1 Setting the DataFactory class

There are multiple possible solutions.

6.1.1 Data Source Configuration

The most simple solution is to set the class name of the DataFactory class in the Data Source Configuration on which the report is based.

Note: A data source configuration with the same name can have different underlying data sources on the server machine and the machine running i-net Designer.

Possible problems with this approach could be that:

- an administrator changes, renames or deletes the data source configuration pointing to your DataFactory implementation
- the DataFactory class can not be loaded

In these cases i-net Clear Reports does its best to locate the data source via fallback solutions.

6.1.2 API

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

Another solution is to set your DataFactory class via API. You need access to the Engine instance after loading. To get the Engine instance you need to use one of the other described programming techniques.

```
Engine engine = ...
DatabaseTables dbt = engine.getDatabaseTables();
for(int i=;i<dbt.getDatasourceCount();i++) {
    Datasource ds = dbt.getDatasource(i);
    ds.setDataFactory(myDataFactory);
}
```

The same must be repeated for each sub-report.

6.2 Implementation of your own DataFactory

If you want to use your own DataFactory class that can be used for report design in i-net Designer and report execution then you need to implement your own DataFactory class that extends from com.inet.report.database.BaseDataFactory (if non JDBC data are used) or com.inet.report.Database (if customization for a used JDBC driver is needed). More details about the methods that need to be implemented can be found in the API documentation.

For more details about the creation of DataFactory class please refer to the sample "DatabaseClassUsableInDesigner" in the directory "documentation/developer/samples/jdbc" of the i-net Clear Reports installation.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

7 Dummy JDBC Driver

If you want set the data via API you can also use your own JDBC driver. JDBC is the standard data factory of Java. At first glance, this approach may appear very difficult because a JDBC driver has an extensive API. But i-net Clear Reports only uses a few of these API methods. You can find a sample for a dummy JDBC driver in the directory: "<installation directory>/documentation/developer/samples/jdbc_your_own_driver".

You can set the utilized JDBC driver in the Data Source Configuration used by your reports. The JDBC driver can be used at design time (in the report designer) and at runtime in the server.

7.1 Building your own JDBC driver

Look at the samples in the folder "<installation directory>/documentation/developer/samples/jdbc_your_own_driver". Copy the package "dummydriver" in your Java IDE and change the implementation of the implemented methods. Create a jar file and add it to the lib directory of the i-net Designer.

Now you can design a report using your own JDBC driver.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

8 EngineRenderData

If you want to implement an application without web interface or your own client-server communication protocol (for example CORBA or SOAP), the recommended solution is to extend the class RenderData of the Java Viewer (client side) and the class EngineRenderData on the server side. The most important method is createEngine(Properties). This is the factory method for the Engine. You need to transfer all method calls from the RenderData on client and back. This class does not use the cache. The Engine and the rendered data is hold in the memory. The i-net DesignerXML uses this class to generate the data for its embedded report viewer.

You can:

- use all features of the Java viewer
- have full control over the Engine creation
- have full control over the client server communication

You can not:

- use the Remote Configuration Manager
- create reports larger than the RAM allows, since the report results are held in memory.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

9 CacheRenderData

The CacheRenderData is an alternative implementation of the RenderData interface. The difference to the EngineRenderData is that it uses the cache. Therefore you can't create the Engine itself.

You can:

- use all features of the Java viewer
- have full control over the Engine creation
- have full control over the client server communication
- create reports larger than memory-only

You can not:

- use the Remote Configuration Manager

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

10 RDC and Engine

The classes com.inet.report.RDC and com.inet.report.Engine are the starting points if you want to work with the low-level API directly. If you use the low-level API then you will have to do many things like caching and client server communication yourself.

You can:

- fully control the Engine creation
- fully control the client server communication

You can not:

- use the Remote Configuration Manager
- create reports larger than your RAM can hold
- use all features of the viewer (search, error messages).

10.1 Units in RDC

10.1.1 Twips, Inches and Millimeters

Twip (twentieth of a point) is a screen-independent unit to ensure that the proportion and position of screen elements are the same on all graphical display systems. A twip is equal to a 20th of a printer's point.

10.1.2 Conversion

- 1 pica = 1/6 inch
- 1 point = 1/12 pica
- 1 twip = 1/20 point or 20 twips = 1 point
- 1 twip = 1/567 centimeter or 567 twips = 1 centimeter
- 1 twip = 1/1440 inch or 1440 twips = 1 inch

The number of twips per pixel depends on hardware and screen resolution (e.g. 800x600: approx. 15 twips per pixel).

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

11 Compare the programming techniques

	Cache	EngineFactory	PropertyChecker	DataFactory
All Viewer Features [1)]	No	Yes	Yes	Yes
Report Cache	Yes	Yes	Yes	Yes
Full control over the Engine creation	limited [2)]	Yes	No	No
Modify report parameters	Yes	No	Yes	No
Modify HTML page	Yes	No	Yes	No
Full control over the client server communication	Yes	No	No	No
Complexity (1 – simple, 5 - difficult)	3	2	1	1
Modify the Engine after loading	limited [3)]	Yes	Yes	No
Set the report data	No	limited [4)]	limited [5)]	Yes

	Own JDBC Driver	EngineRenderData	CacheRenderData	RDC and Engine
All Viewer Features [6)]	Yes	Yes	Yes	No
Report Cache	Yes	No	Yes	No
Full control over the Engine creation	No	Yes	No	Yes
Modify report parameters	No	Yes	Yes	Yes
Modify HTML page	No	Yes	Yes	Yes
Full control over the client server communication	No	Yes	Yes	Yes
Complexity (1 – simple, 5 - difficult)	3	4	4	5
Modify the Engine after loading	No	Yes	No	Yes
Set the report data	Yes	limited [7)]	No	limited [8)]

Our recommendation is to use ReportServlet, EngineFactory or both together. If you only want to set the data then your own Database class is the best solution.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

12 Formula Expander Class(es) - Formula Functions

It is possible to use your own functions in a formula of a report, if these functions are implemented as methods in public (formula expander) Java class(es). To use such "user defined functions" in a formula, the following steps are required:

- Best practice is to write a plugin and to register your classes with the marker interface com.inet.report.formula.UserDefinedFunction in the ServerPluginManager. Details on writing a plugin can be found in the chapter "Plugins".
- Alternatively you can use an old school jar file:
    - Copy this jar file into the "lib" directory in the installation directory of i-net Clear Reports.
    - Open the Configuration Manager and add the `<classname>` to the property 'Formula Expander Class(es)', where `classname` means the fully qualified name of your class. This property can as well be a semicolon separated list of class names. Please check the classpath used by i-net Clear Reports and the package structure in case ClassNotFoundException occurs at runtime. Alternatively you can use the API to modify the configuration in use.
- Now you can write a formula using your user defined functions by calling them by name. You can create a formula using i-net Designer or i-net Clear Reports API (RDC).
- Note 1: The methods in the specified class have to be public and could be static. In case of non-static methods, a default Constructor without parameters is required.
- Note 2: The return value and the parameter values of the methods have to be of the following types:
    - com.inet.report.FormulaRange
    - java.lang.Number
    - java.lang.Boolean
    - java.lang.String
    - java.sql.Date
    - java.sql.Time
    - java.sql.Timestamp
    - Object[] of these types

As of version 13 it is possible to use one or more of the following hidden parameters as method parameter in the formula expander class: "Engine",

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

"HttpSession" and "HttpServletRequest". This can be useful, for example, to use the Engine API to get user defined parameters from the report engine. Please note that "HttpServletRequest" does not work in the i-net Designer.

12.1 Examples:

```
static public String aFunction(String str) {
 return "[" + str + "]";
}

static public String getUserProperty(String propKey, Engine engine) {
        Properties properties;
 String userProperty = "";
        try {
            properties = engine.getUserProperties();
            if (properties != null) {
                userProperty = (String)properties.get(propertyKey);
            }
        } catch( ReportException e ) {
            e.printStackTrace();
        }
        return userProperty;
}

static public Number aFunction(Number d) {
 return new Double( 2.0 * d.doubleValue());
}

static public String aFunctionToo( Object obj) {
 if (obj instanceof String) {
  return ((String) obj) + " Ha";
 } else {
  return null;
 }
}

static public Number aFunction(Number i) {
 return new Integer(i.intValue() * 2);
```


© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

```
}

public Object[] aFunction(Object[] i){
 Object[] o = new Object[i.length];
 for (int k=;k<i.length;k++) {
  o[k] = i[i.length-k-1];
 }
 return o;
}

public Boolean aFunction( Number i, FormulaRange range ){
 double lower = ((Number)range.getFrom()).doubleValue();
 double upper = ((Number)range.getTo()).doubleValue();
 double n = i.doubleValue();

 if( range.isLowLimitIncluded() ? n < lower : n <= lower ){
  return Boolean.FALSE;
 }
 if( range.isHighLimitIncluded() ? n > upper : n >= upper ){
  return Boolean.FALSE;
 }
  return Boolean.TRUE;
 }
}

// Note: This function should do a type check on the range bounds first.
```

You can find more samples in the Java code samples.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

13 Plugins

If you want to add a new feature to i-net Clear Reports, then it is recommended to create a plugin. A plugin is a zip file in the folder "plugins" which contains all required files for this new feature.

In order to remove a feature from your i-net Clear Reports installation, just remove its plugin zip file and restart the server. If another plugin requires the removed plugin, this plugin will be disabled. A user can also disable a non core plugins with the remote GUI.

13.1 Architecture

A plugin is a zip file. It contains at least the files plugin.properties and server.jar in order to work.

The **plugin.properties** defines the start class (named **entrypoint**) which implements the interface com.inet.plugin.ServerPlugin. All dependencies, id, description and included internal jar libraries must be defined in this file.

The **server.jar** contains the code of your plugin. At least an implementation of the interface com.inet.plugin.ServerPlugin is required.

13.2 Live cycle

- On start-up, i-net Clear Reports loads all plugins that are available in the **plugins** directory.
- All located plugins are opened and their server.jar is extracted into a temporary directory.
- The **plugin.properties** files of the plugins are read and a dependency tree is created.
- If the dependency of a plugin is not fulfilled, it is disabled.
- Now each plugin gets its own classloader which has access to the plugin classes, internal and external referenced libraries and classes that are made public by other plugins (if the current plugin has an appropriate dependency defined).
- The next step is to instantiate the **entrypoint** class.
- The **registerExtension** method is called. Each plugin may now register instances of

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

extensions it provides. This phase is meant only for registration, initialization is discouraged here. As a result, it is not allowed to query the ServerPluginManager for already registered instances. This call would fail with an exception. If a single plugin is throw an exception then the server is stopped.

- After the **register** phase, the **init** phase is started which calls the **init** method of your plugin **entrypoint** instance. Now all required instances can be initialized. Since all extensions are registered now, it's allowed to fetch the registered extensions from the provided ServerPluginManager instance. Registering further extensions is disabled in this phase and will result in an exception.

If background threads are needed, they can be started in the **init** method. Please ensure to stop these threads in the **reset** method. A restart of the background thread should be performed if the **restart** method is invoked. Resetting and restarting is only invoked by some application servers that support unloading unused web containers.

13.3 Plugin Properties

Out of the following list, only **id** and **entrypoint** are mandatory. Any further property is optional.

Property	Description
id	A unique identifier with only lowercase characters. In general it is equal to the filename of the plugin.
entrypoint	The fully qualified class name of the server plugin entry class. Packages are separated with dots.
dependencies	A semi-colon separated list of required plugins identified by their ID (optionally with comma-separated version).
optionalDependencies	A list of optional dependencies. The current plugin will run even if these dependencies are missing. It give access to the exported classpath of this plugins. You need to check the availability with isPluginLoaded(id).
internal	An optional semi-colon separated list of additional jar files in the zip file. These jar files will be added to the classpath of the plugin.
external	Any external library jar files required by this plugin. A semi-colon separated list with absolute paths or paths relative to the current working directory.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

Property	Description
version	The version number of this plugin. It will be shown in the log file. It has to be a valid float number and can also be used to define the dependency to a plugin with a special version.
description	A description that will be shown in the GUI.
description_<locale>	A localized description that will be shown in the GUI.
initAfter	A semi-colon separated list of dependencies identified by their ID. The plugin manager will try to init any of the plugins in this list before the current one. In case of a cyclic reference the order in this cycle is undefined.
flags	A semi-colon separated list of flags: * **designer** - also loaded from remote designer * **rootAccess** - If the plugin need access to the root class loader of the application. A setting of the dependencies property is to prefer. * **core** - is hidden from the plugin manager in the configuration manager GUI and can not be disabled * **optional** - plugin is not enabled by default. The user must be enable it in the configuration manager GUI. * **''** - empty string: enabled by default and visible in plugin manager

13.4 Extension points

i-net Clear Reports has a growing list of extension points. Extension points are interfaces or abstract classes that a plugin can implement to extend the functionally of another plugins or the core. Here is a short list of available extension points.

Extension	Plugin	Description
com.inet.report.schedule.ScheduleAction	scheduler	Additional action for the scheduler.
com.inet.http.PluginServlet	-	Adds an extra servlet context to the server.
com.inet.authentication.UsersAndGroupsProvider	-	Provides a list of available users and groups for the security settings in the GUI.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.

Extension	Plugin	Description
com.inet.authentication.AuthenticationProvider	-	Add support for an additional authentication mechanism.
com.inet.report.formula.UserDefinedFunction	-	Add custom formula functions.
com.inet.report.database.DataFactory	-	A data factory which is not in lib.
com.inet.config.structure.provider.ConfigStructureProvider	-	Adds extra dialogs to configuration manager.
com.inet.report.encode.DecoderFactory	-	Adds support for decoding of image and/or document types.

13.5 Sample Code

Samples for a scheduler action and a user-group-provider including a step by step description can be found in the samples folder of your i-net Clear Reports installation.

1) , 6) The error handling and the search feature is not always possible
2) , 3) With Cache.addEngine(Engine) no large reports are possible
4) , 5) , 7) , 8) You can not set the data for each individual sub report.

© Copyright 1996 - 2015, i-net software GmbH; All rights Reserved.