

Application Designer

About this Documentation

Version 9.1.2

October 2019

This document applies to Application Designer Version 9.1.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2005-2019 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: CIT-MULTILANG-912-20190816

Table of Contents

1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Multi Language Management	5
3 Writing Multi Language Layouts	7
Example	8
Page Name Strategy	9
4 Creating the Translation File	11
5 Defining the Language at Runtime	13
6 Dealing with Literals inside Your Adapter	15
7 Tools for Translating Text IDs	17
8 Tool for Creating Languages	19
9 Unicode	21
10 Interface IMLManager	23

1

About this Documentation

■ Document Conventions	2
■ Online Information and Support	2
■ Data Protection	3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Multi Language Management

The multi language management is responsible for changing the text IDs into strings that are presented to the user.

There are two translation aspects:

- All literals in the GUI definitions of a layout are replaced by strings which are language-specific.
- Literals you output within your adapter code (e.g. status messages) must be translated.

The multi language management is internally kept cleanly behind an internal interface. This means that in the future a different implementation will be available to provide a solution to find a string for a given text ID. In this documentation, the default implementation which simply uses comma separated value files is described.

The information provided in this documentation is organized under the following headings:

[Writing Multi Language Layouts](#)

[Creating the Translation File](#)

[Defining the Language at Runtime](#)

[Dealing with Literals inside Your Adapter](#)

[Tools for Translating Text IDs](#)

[Tool for Creating Languages](#)

[Unicode](#)

[Interface IMLManager](#)

3

Writing Multi Language Layouts

■ Example	8
■ Page Name Strategy	9

When defining properties of controls inside a layout definition, there are always two options to specify fix labels: either use property `name` or property `textid`. In case your pages support multi language ability, you only have to use the `textid` property. At runtime, the corresponding labels are found in the following way:

- Each PAGE has the property `translationreference`. This property may be the name of the HTML file - or it may be a logical name, used by different HTML pages.
- Inside Application Designer, there are defined directories and files in which the text information is stored: each application project is represented by a directory under the web application directory of Application Designer. (See *Application Project Management* for more details on managing projects.) Inside the project directory, there is a directory `/multilanguage/`. Under this directory, each language is represented by its own directory, e.g. by the directory `/multilanguage/de/` for German translations.
- Inside each language directory, there is one comma separated value (CSV) file for each page name. The name of the file is `<pagename>.csv` (for example, `Login.csv`).
- Inside the CSV file, each line contains the text ID, a semicolon and the label text, e.g. "Label1;Login name".

Example

Let us assume you have defined an application project "accountmgmt". Inside the application project, there is a layout definition `account.xml` that points via the `translationreference` property of PAGE to "account". The file structure inside your application project directory now looks as follows:

```
<webapp-directory>/
  accountmgmt/
    account.html           // generated HTML file
    multilanguage/
      de/
        account.csv       // German text
      en/
        account.csv       // English text
    xml/
      account.xml         // layout definition
```

Page Name Strategy

The previous section explained how a translation file is found for a certain HTML page. Basically, the translation reference is used to link the layout definition and the Application Designer multi language management.

In general, there are two strategies for using this translation reference, and a mixture of both:

- Specify one central page name for a couple of pages. Therefore, all pages share the same multi language information (i.e. the same *.csv* file).
- Specify one page name for each page. Therefore, every page has its own *.csv* file.

For larger projects, it makes sense to combine different literal information into one file - in order to keep consistency and to avoid redundancy. Of course, you have to synchronize the naming of text IDs for each page.

4 Creating the Translation File

The translation file (*account.csv* in the [example](#) of the previous section) is a simple comma separated file with the following format:

```
textid1;text1  
textid2;text2  
textid3;text3
```

If your text itself contains a semicolon, then write "\;".

You can either create the file by using a text editor or you can use Application Designer's Literal Assistant which is integrated in the Layout Painter.

Pay attention: when using text editors of your own, you must configure your editor to store the text using UTF-8 character encoding. Otherwise, any characters that are not "ASCII characters < 128" will not be properly displayed. Make sure that your editor is UTF-8 capable.

5 Defining the Language at Runtime

With the protected member `m_sessionContext`, you find or set the currently active language used by the multi language management:

```
...  
m_sessionContext.setLanguage("de");  
...
```

The value passed to the session context is only valid within the context of current session. Therefore, different users can be logged on to the system choosing different languages.

The string, which is passed to the `setLanguage()` method of `m_sessionContext`, represents the name of the directory in which the CSV files are stored. You are not bound to the "de" and "en" directories; you can add any other directories representing additional languages.

6 Dealing with Literals inside Your Adapter

Use method `replaceLiteral` of the inherited Adapter class to replace messages:

```
...
this.outputMessage("S",replaceLiteral("APP1","successFileSaved"));
...
```

The first parameter is an abbreviation - the file name of the multi language file. The second parameter is the text ID that should be translated into text as described previously.

It is also possible to pass parameters of your application to the multi language management. For example, if you want to show a success message which informs that file "xyz" was saved, proceed as follows:

```
...
String fileName = "xyz";
this.outputMessage("S",replaceLiteral("APP1","successFileSaved",fileName));
...
```

The corresponding line in the CSV file (*APP1.csv*) in the *\en* directory for English looks like:

```
...
successFileSave;File &1 was saved successfully
...
```

The "&1" is automatically replaced with the file name. There are other variants of the `replaceLiteral()` method available to pass 2 or 3 parameters. In this case, use &1, &2 and &3 in the text definition.

7

Tools for Translating Text IDs

There are two tools. One is the Literal Assistant that is part of the Layout Painter. The other is the Literal Translator.

For detailed information on these tools, see

- *Using the Literal Assistant and Literal Translator in the Development Workplace* documentation.
- *Using the Literal Assistant and Literal Translator in the Ajax Developer* documentation.

8 Tool for Creating Languages

Application Designer comes with two languages: "en" for English and "de" for German. When creating a new language abbreviation, you have to take care of the following:

- You have to create language directories in your projects.
- You have to copy certain files in which Application Designer holds text information that is language dependent.

The Language Manager automates the creation of language abbreviations. For detailed information on this tool, see *Language Manager* in the *Development Workplace* documentation.

9 Unicode

Pay attention to the fact that Application Designer is fully based on Unicode and its UTF-8 format. All multi language files must be in UTF-8 format. Especially pay attention when maintaining CSV files with programs like MS Excel.

10

Interface IMLManager

Application Designer uses CSV files as the default implementation for storing translation information. In some scenarios, you do not want Application Designer to be responsible for storing this information, but want to store translation information on your own.

Example: you may already have a text database within your existing application, and you may already have defined procedures on top of this: translation tools, automated translation etc.

Application Designer provides an interface `com.softwareag.cis.multilanguage.IMLManager` that is internally used for accessing translation information:

```
public interface IMLManager
{
    public void refreshBuffers();
    public String getString(String language,
                           String project,
                           String application,
                           String literal);
    public boolean checkIfExists(String language,
                                String project,
                                String application,
                                String literal);
    public String getString(String language,
                           String project,
                           String application,
                           String literal,
                           String[] embeddedStrings);
    public String[] getApplicationStrings(String language,
                                           String project,
                                           String application,
                                           String[] textIds);
}
```

See the JavaDoc API documentation for detailed information.

You can implement this interface by a class of your own and register this class in the Application Designer runtime environment. The registration is done inside the *cisconfig.xml* file:

```
<cisconfig ...  
    multilanguagemanager="<your implementation>"  
    ...>  
  
↵
```

Be aware of the fact that the interface is a pure “access interface” that is used at runtime; i.e. whenever Application Designer requires information about a certain text ID, the interface is called. The interface currently does not provide methods for writing text information back, i.e. there is no coupling of Application Designer's multi language tools (e.g. translation inside the Layout Painter).