

Kofax FraudOne

Java Client Customization Layer

Version: 4.5.0

Date: 2020-11-04

The KOFAX logo is displayed in a bold, blue, sans-serif font. The letters are thick and closely spaced, with a consistent weight throughout the word.

© 2020 Kofax. All rights reserved.

Kofax is a trademark of Kofax, Inc., registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Kofax.

Table of Contents

Preface	4
Getting help with Kofax products.....	4
Training.....	4
Related documentation.....	5
Chapter 1: Introduction to the customization layer	6
Purpose of the customization layer.....	6
Customization features provided by this customization layer.....	6
Chapter 2: Unified configuration interface	7
Configuration string compatible with java.util.ResourceBundle.....	7
Other configuration functionalities.....	7
Chapter 3: GUI customization guideline	8
General guidelines.....	8
EasyExtension.....	9
Supported panels/dialogues.....	9
The extension widgets.....	10
Chapter 4: Logon process customization	11
Turn on extensible logon module.....	11
Different module options on logon.....	11
Logon module implementation interface.....	11
Chapter 5: Reference	13
Chapter 6: Supported extension specification	14

Preface

Getting help with Kofax products

The [Kofax Knowledge Base](#) repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base, go to the [Kofax website](#) and select **Support** on the home page.

Note The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.
Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.
Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).
Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).
Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.
- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.
Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Training

Kofax offers both classroom and computer-based training that will help you make the most of your Kofax FraudOne solution. Visit the Kofax website at www.kofax.com for complete details about the available training options and schedules.

Related documentation

The full documentation set for Kofax FraudOne is available at the following location:

<https://docshield.kofax.com/Portal/Products/FO/4.5.0-th2k87ey6r/FO.htm>

In addition to this guide, the documentation set includes the following items:

Guides

- *Kofax FraudOne Administrator's Guide*
- *Kofax FraudOne Data Warehouse Installation and Operation Guide*
- *Kofax FraudOne Extended Reporting Features and Statistics*
- *Kofax FraudOne Feature Codes*
- *Kofax FraudOne Installation and Migration Guide*
- *Kofax FraudOne Java Client Customization Guide*
- *Kofax FraudOne License Management*
- *Kofax FraudOne Report Component Installation Guide*
- *Kofax FraudOne SignCheck Result Codes*
- *Kofax FraudOne Standard Reporting Features and Statistic*
- *Kofax FraudOne The Book on CRS*
- *Kofax FraudOne Thin Client Customization Guide*
- *Kofax FraudOne Thin Client Customization Layer*

Interfaces

- *Kofax FraudOne Archive Interface Server*
- *Kofax FraudOne ASV Blackbox*
- *Kofax FraudOne Global Fraud Signature Web Service Developer's Guide*
- *Kofax FraudOne Service Program Interfaces*
- *Kofax FraudOne User Login Procedure*
- *Kofax FraudOne Standard Teller Interface*
- *Kofax FraudOne Variant Cleanup Utility*

Online Help

- *Kofax FraudOne Administration Client Help*
- *Kofax FraudOne Java Client Help*
- *Kofax FraudOne Server Monitor Help*
- *Kofax FraudOne Thin Client Help*

Chapter 1

Introduction to the customization layer

This section briefly introduce the purpose of the customization layer and customization possibilities using this customization layer.

The reader may need read other FraudOne support documents, e.g. *Kofax FraudOne Administrator's Guide*, to be familiar with the basic FraudOne configuration.

Purpose of the customization layer

Two main purposes of this customization layer are:

- To unify the configuration interface for SignBase module and SignCheck module
- To make it easier to customize user interface, through the so called EasyExtensions mechanism

Customization features provided by this customization layer

With this new customization layer, user can:

- Use those special configuration features from SignCheck in configurations for SignBase module
- Customize the UI for displaying in a much easier manner
- Other minor customization options, including the extensible logon module and logging system (through XjLog, which is based on logback)

The logback logging is activated by creating a standard Logback configuration file under the same directory as the Java Client files.

Chapter 2

Unified configuration interface

Configuration string compatible with java.util.ResourceBundle

The configuration system is based on `java.util.ResourceBundle`, thus all the configuration compatible with `ResourceBundle`.

Most of the time, the configuration string contains a key or parameter (the string on the left of the "=" and is compulsory) and a value (the string on the right of the "=", and is optional).

Example

```
First.var=val
```

Here, the key is "First.var", and the value for "First.var" is "val".

Other configuration functionalities

Other configuration functionalities are:

- Specify value in another properties file by using asterisks forward (*custom*)
- BNO specific properties by using `bnoproducts` configuration
- Classpath lookup of properties (properties file lookup is done in order: 1. core, 2. custom.zip, 3. classpath)
- Caching of properties (PropertyHandler: configuration is hold in HashMap)
- Formulas and functions (see *Kofax FraudOne Java Client Customization Guide*)

Keys, that are marked with (f), can be defined with a formula:

```
${<key><delm><function><delm><function...>
```

where key is a key from the hashtable, delm is one of the characters "|", "?" or ":" and function one of the defined functions, analogous to the formulas in the table properties files.

Example

```
deleteDataFile =${BNO|TEST$*<=305?FMT0:FMT1}:
```

- Chaining of properties functionality (see *Kofax FraudOne Java Client Customization Guide*)

There are some special keys that allow to take over keys from another properties file or a whole properties file in the current properties file.

Chapter 3

GUI customization guideline

General guidelines

Extension data are additional information needed depending on customer's actual requirement. Supported extension can be found in appendix 6.1. They are also listed below.

- Char
- Char-Binary
- SmallInt
- Integer
- Decimal
- Date
- String

Other required information to specify an extension data includes ID, Relation, and Default (optional)

Each item of extension data is identified uniquely by a string. For a given item of extension data, there is a corresponding table entry in DB side and one or several corresponding UI components to actually interact with the table entry, including initialization, modification and display. For type of widgets currently supported, refer to section 3.2.1 of the PS customization layer design document.

The configuration information needed to store the extension data in DB can be specified in SignBase.properties, by following the steps below:

Step 1: Specify the necessary information required for the extension, call it ext1, by prefix Type, ID, Relation, and Default with the extension name, i.e. ext1 here

```
ExtensionSpecification. ext1.ID          = <unique ID, among extensions of the same  
relation>  
ExtensionSpecification. ext1.Type       = <supported type>  
ExtensionSpecification. ext1.Relation   = <predefined relation>  
ExtensionSpecification. ext1.Default    = <default value>
```

Step 2: Add the extension name to a built in key, called ExtensionSpecification.Extensions, delimited by a SPACE.

```
ExtensionSpecification.Extensions = <existing extension names> ext1
```


EasyExtension

Via easyExtension mechanism, customization involving extension handling can be handled easily through configuration, ideally.

EasyExtensions are currently available for those dialogs/panels required by the design document, section 3.2.1. Each of these dialogs/panels are shown as a separate panel, in which the extensions are placed in ONE row only, either vertically (except customer information panel) or horizontally (customer information panel), one after the other. There locations are fixed, for now, as specified in design document. This separate panel will have a configurable titled border and is scrollable to support the display of unlimited extensions.

Supported panels/dialogues

Supported panels are grouped based on the content they are containing, and the panels/dialogues within each group are interacting with same set of extensions. Thus the extensions for these panels/dialogues are specified in signbase.properties using a key with common fix. Below lists the relationship between the display information and common fix.

To display information for	Common fix used
Customer information (customer model)	SBAM_Customer
Customer information (account model)	SBAM_CustomerAccount
Account information	SBAM_Account
Signatory information	SBAM_Signatory
Account attachment	SBAM_AccountAttachmentDetails
Check stock image	SBAM_CheckstockImageDetails
Signatory rule	SBAM_Rule
Quick information on customer and account	SBAM_CustomerInfoPane

Configurable options available at extension panel/dialogue level:

Option	Description
<commonFix>.Extensions	Extension names separated by space
<commonFix>.Extensions.Border.Visible	Visibility of the extension border
<commonFix>.Extensions.Border.Title	Title of the extension border
<commonFix>.Extensions.Dimension.Height	Default height of the extension panel
<commonFix>.Extensions.Dimension.Width	Default width of the extension panel

Here, commonFix refers to any of above listed common fix. Here is an example from design document, for SBAM_Customer.

```
SBAM_Customer.Extensions = ExtName1 ExtName2 ExtName3 ExtName4
SBAM_Customer.Extensions.Border.Visible = 1
SBAM_Customer.Extensions.Border.Title = *language*extensions.border.title
```

```
SBAM_Customer.Extensions.Dimension.Height =
SBAM_Customer.Extensions.Dimension.Width =
```

Specially for SBAM_CustomerAccount, the corresponding extended fields are following that for Account, e.g.

```
ExtensionSpecification. <extension name>.Relation = A
```

The extension widgets

Each widget actually comes with a label, by default, on its left to provide relevant information on the data contained in this widget.

The following configurations are available for an extension widget within each extension panel/dialogue.

Configuration	Description
<commonFix>.ExtName1.Label	Label
<commonFix>.ExtName1.Size	Capacity of the widget
<commonFix>.ExtName1.MinSize	Required minimal size on input data
<commonFix>.ExtName1.Visible	Visibility of this widget
<commonFix>.ExtName1.Enabled	If this widget is enabled
<commonFix>.ExtName1.Mandatory	If this widget is mandatory
<commonFix>.ExtName1.Mask	Mask, if needed
<commonFix>.ExtName1.AutomaticFocusTraversal	If automatic focus traversal is enabled
<commonFix>.ExtName1.ConvertToUppercase	If conversion to uppercase is needed, by default
<commonFix>.ExtName1.Widget	Type of the widget- 1 - text field 2 - text area 3 - combo box 4 - checkbox

Here is an example:

```
SBAM_Customer.ExtName1.Label = *language*extName1.Label
SBAM_Customer.ExtName1.Size = *sizes*customer.extName1.size
SBAM_Customer.ExtName1.MinSize = *sizes*customer.extName1.minSize
SBAM_Customer.ExtName1.Visible = 1
SBAM_Customer.ExtName1.Enabled = 1
SBAM_Customer.ExtName1.Mandatory = 1
SBAM_Customer.ExtName1.Mask =
SBAM_Customer.ExtName1.AutomaticFocusTraversal =
SBAM_Customer.ExtName1.ConvertToUppercase =
SBAM_Customer.ExtName1.Widget = 1
```

Chapter 4

Logon process customization

The logon module is one of the modules that requires customized frequently, especially through LDAP. The customization layer abstract out the logon module into a separate module, making it easier to customize the logon process.

Turn on extensible logon module

To turn on the new extensible logon process, set

```
SB_Workspace.usingLogonModule = 1
```

in `custom.properties`.

Once turned on, FraudOne will examine different logon module options described below and logon user accordingly.

Different module options on logon

Besides those built in logon approaches, it is possible to implement customized logon approach to meet customer's requirement flexibly.

If turned on, the logon module will initialize an object to handle the logon process, in the following order:

1. If the `LogOnModule.ClassName` is defined in `SignBase.properties`, it will try to initialize an instance of the specified class.
The customized logon class shall be implemented as the way described below.
2. If the `LDAP.Enabled = 1` in `SignBase.properties`, it will try to logon using the built in LDAP logon module
3. For all other cases, including exception is caught from above two case, the default logon module is used.

Logon module implementation interface

Any customized logon module must meet the following requirements:

- It extends the abstract class `LogOnModule`

- It implements the following functions:
 - `public SP_User logonUser(String userId, byte[] pwd, byte[] newPwd, int p_logonFlag) throws SP_Exception;`
 - `public void doLogoffUser(String clientName, byte[] ipAddress) throws SP_Exception;`

Chapter 5

Reference

1. *Logging and Tracing with XLog*
2. *Kofax FraudOne - Java Client Customization Layer*

Chapter 6

Supported extension specification

The existing way of specifying extension relationship can be found in `signbase.properties`. Here is a brief introduction.

To specify extensions

```
ExtensionSpecification.Extensions = ExtName1 ExtName2 ...
ExtensionSpecification.ExtName1.ID = <ID used to store the extension in database>
ExtensionSpecification.ExtName1.Type = <the type identifier (0-7)>
ExtensionSpecification.ExtName1.Relation = <the object type the relation belongs to>
ExtensionSpecification.ExtName1.Default = <the default value used when an extension is
created>
```

Supported extension type

Type identifier (0-7)	Data type
1	Char
2	Char-Binary
3	SmallInt
4	Integer
5	Decimal
6	Date
7	String

Supported object type

Relationship char	Bond to
C	Customer
A	Account
Y	Signatory
G	Groupin
S	Signature
R	Rule
L	Rule_Group
I	Account_Image
D	Document
K	Mask

Relationship char	Bond to
U	User
P	StockImage