

Kofax FraudOne

Administrator's Guide

Version: 4.4.2

Date: 31-Jan-2020

KOFAX

© 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

Chapter 1 Preface	12
Related documentation	13
Training	14
Get help for Kofax products	14
Chapter 2 FraudOne architecture	15
Overview FraudOne architecture	15
Server Manager	16
Attach Manager	17
FraudOne interfaces	17
FraudOne databases	18
SignBase	18
SignInfo	19
SignCheck	19
Data Warehouse and Archive	20
Application Server layer	20
SignBase Server	21
SignCheck Server	21
FraudOne Engine Server	22
SignCheck Workflow Router	22
FraudOne visual clients	22
Java Client	23
Thin Client	23
Administration Client	24
FraudOne concepts	24
FraudOne Data Model	24
Historisation	26
Variants	26
Four Eyes Principle	26
Authorization models	27
Multiple bank support	28
Chapter 3 FraudOne databases	28
General	28
Overview FraudOne databases	28
Prerequisites and consideration.....	29
SignBase database	29
SignCheck database	30
Creating and maintaining the databases	30
General information for all databases	30
Database administration.....	30
Schema names.....	31
Codepages and national language characters.....	31
Allocation of disk space.....	31
Authorization.....	31

Trusted connection.....	32
Table creation programs.....	32
SignBase table names.....	32
SignCheck table names.....	33
Command options.....	33
Property file options.....	34
DB2 database	38
Database code page.....	38
After the creation of the databases.....	39
Authentication and authorization.....	39
Maintenance of the database.....	39
ODBC configuration.....	40
DB2 on IBM mainframe systems.....	40
Oracle database	40
ODBC settings.....	40
Database code page.....	40
Number of open cursors.....	40
Authentication and authorization.....	41
Table space.....	42
Maintenance of the database.....	42
Oracle ODBC settings.....	42
Microsoft SQL Server database	42
Database code page.....	42
Disk space for the database.....	43
Schema names.....	43
Authentication and autorisation.....	43
SQL Server ODBC settings.....	44
SignBase and SignCheck database validation	44
Data Warehouse	47
Schema names	47
Preparing the data structures	47
Configuration entries for all databases.....	47
Configuration entries for DB2 Windows / Unix / Linux.....	48
Configuration entries for DB2 Z/OS.....	49
Configuration entries for SQL Server.....	53
Configuration entries for Oracle.....	54
Starting the installation.....	55
Chapter 4 FraudOne Server Manager	55
Overview FraudOne Server Manager	55
Server Manager, Attach Manager and FraudOne Servers	56
Server Manager installation	56
Server Manager configuration	57
SetupCfg.cmd command options for client installation	57
SetupCfg.cmd command options for configuration server installation	58
SetupCfg.cmd command options for other server installation	59
SetupCfg.cmd command options to update the database technical user and password	59
SetupCfg.cmd command options for remote installation	60
SetupCfg.cmd command options for configuration management	60
Using the Server Manager	61
Starting Server Manager	61
Starting Server Manager as a Windows Service	62
Command line options for the Server Manager	63
Scheduling the FraudOne Application Servers	63
Starting and stopping manually.....	63

Starting and stopping automatically	63
FraudOne Scheduler (Windows only)	64
FraudOne Scheduler File Format Specifications.....	64
XML Sample Based on the DTD File Format.....	65
Supported entries.....	65
DTD file format specifications.....	67
Administration with the Server Monitor	68
Attach Manager	68
Attach Manager installation	68
Attach Manager configuration	68
Attach Manager communication protocol	69
Timeouts for the different clients.....	69
Timeouts for visual clients (Java Client, Thin Client, Administration Client)	70
Timeouts for automatic clients (Engines: ASV, Check Stock, etc.).....	70
Message type lists.....	71
Centralized configuration for SignCheck components	71
SignBase Application Server configuration	71
Support for multiple SignBase Databases	71
Machine names	72
Configuration Server addresses	72
Configuration setup program	72
Server Manager configuration file elements for SrvMngr4.ini	72
Settings in the Server Manager configuration file	73
Server Monitor and Manager [Monitor]	73
SignBase Application Server	74
SignBase Attach Manager	75
SignCheck Application Server	75
SignCheck Attach Manager	75
Archive Interface Server	76
Workflow Router	76
CRS Engine	76
Schedule Server [SchedSrv]	77
ASV Automat Attach Manager	77
ASV Automat Server	77
APSV Attach Manager	78
APSV Server	78
APSV (getNext) Server	79
SDQ Attach Manager	79
SDQ Server	80
ICV Server	80
GIA Attach Manager	81
GIA Server	81
Service Programs	82
Chapter 5 FraudOne Administration Client	82
General	82
FraudOne Administration Client standard functionality	82
FraudOne Administration Client configuration.....	82
FraudOne Administration Client requirements.....	83
Administrating with the Application Server configuration file elements.....	83
Administration Client	83
Installing the Administration Client	83
Installing the Administration Client icon.....	84
Initial setup	86

Configuring the Administration Client	86
The admin.properties configuration file element.....	87
Connection parameters.....	87
Trace level.....	88
Default settings.....	89
Property file location.....	89
BNO specific settings.....	90
User Administration settings.....	90
Parameter Administration settings.....	90
CRS Editor settings.....	90
Blacklist Administration settings.....	90
Limitations of the Administration Client	91
Chapter 6 FraudOne clients	91
General	91
Overview FraudOne clients	91
FraudOne clients standard functionality	91
FraudOne client configuration.....	92
FraudOne client requirements.....	93
Administrating with the SrvMngr4.ini configuration file.....	93
FraudOne Java Client	93
Installing the Java Client	94
Installing the Java Client icon.....	94
Initial setup.....	97
Using the Java Client	97
Scan Control Image import	98
Scanning requirements.....	98
Using the Scan Control Image import.....	99
Using the Java Client in Teller Mode.....	99
Help feature.....	99
Configuring the Java Client	99
The custom.properties configuration file.....	99
Connection settings.....	100
Trace level.....	102
Default settings.....	103
Look and feel.....	105
Verification rate settings.....	105
Property file location	105
Scan control settings	106
Queue smulation settings.....	106
BNO specific settings.....	106
Variants settings.....	106
Image Loader settings.....	107
Limitations of the Java Client	107
Teller Interface	107
Installation	107
Usage	108
Configuration	108
Limitations	108
FraudOne Thin Client	108
Installing the Thin Client	108
Installation of Apache (Optional).....	109
Installation of Tomcat.....	110
Installation of the Thin Client components.....	110
Updating a Thin Client installation.....	111

Starting Tomcat as Service.....	112
Web browser configuration.....	112
Manual registration of Thin Client ActiveX Controls.....	114
URI encoding for German Umlauts.....	115
Using the Thin Client	115
Operation Modes.....	115
Standard Thin Client.....	115
Teller Mode.....	115
Application Modes.....	116
Standard Thin Client.....	117
SignInfo Classic Edition (SICE) Client.....	117
Configuring the Thin Client	117
tcustom.properties.....	117
General Thin Client settings.....	117
Thin Client settings for SignBase display.....	118
Thin Client settings for SignCheck.....	119
Teller specific Thin Client settings.....	119
LDAP specific Thin Client settings.....	120
Data Warehouse specific Thin Client settings.....	120
Image Zooming Dimensions and Scaling Strategies.....	121
Plugin Properties	122
Web Server Side Login.....	122
Thin Client Specific Business Model settings in custom.properties.....	124
Standard Login.....	125
Client-side Login.....	125
Check account number.....	126
Logging	126
Static Logging Setup.....	126
Dynamic Logging Setup.....	127
Limitations of the Thin Client	128
Chapter 7 SignCheck Engines	128
General	128
Overview SignCheck Engines	129
Centralized configuration	129
Configuration information	129
Prerequisite configuration actions	130
SignCheck Engines	130
Automatic Signature Verification (ASV)	131
Match Rates ASV.....	132
SIVAL Neural Networks Engine.....	132
Use Only Clean Signature Snippets.....	133
Twin Detection.....	133
Generic Image Analysis (GIA)	133
Match Rates GIA.....	134
Automatic Rule Verification (ARV)	134
Automatic Pad Signature Verification (APSV)	134
SignCheck Engine configuration	135
General configuration with the Administration Client	135
ASV Client [ASVCL-1].....	135
APSV Client [APSVCL-1].....	136
GIA Client [GIACL-1].....	136
The configuration with automat2.ini	137
[Common].....	137
[Automat].....	138

[APSV]	138
[GIA]	138
[BNO]	139
Chapter 8 SignCheck CRS	139
Overview SignCheck CRS	140
System architecture	140
The workflow process	141
Using workflow servers	143
Using CRS	145
Starting and stopping components	145
Supporting tools and utilities	145
WFFlush.....	145
WFDecide.....	146
CRS configuration	147
Workflow Router configuration	147
Configuration for the Server Manager	147
WFRouter.ini configuration	148
CRS configuration	148
Configuration in the Server Manager configuration file	148
CRS.ini configuration (Configuration Server).....	148
Workflow Server configuration	149
Configuration in the Server Manager configuration file.....	149
Workflow API Configuration.....	149
CRS Variables	150
CRS predefined variables	150
CRS customer specific variable configuration	152
Chapter 9 Kofax FraudOne Reports	154
Overview Kofax FraudOne Reports	154
System Architecture of the Kofax FraudOne Reports	154
Reports usage	155
Kofax FraudOne Reports installation	157
Packages	157
Prerequisites	157
Installation	158
Chapter 10 Server Monitor	158
Overview Server Monitor	158
Server Monitor setup	158
Using the Server Monitor	159
Using the embedded Help feature	159
Button bars and screen explanations	159
Authorization	160
Server Monitor configuration	160
Chapter 11 FraudOne SNMP	162
Overview FraudOne SNMP	162
Files used	163
SNMP installation	163

SNMP objects description	163
Using the SNMP Module	164
Chapter 12 FraudOne Service Programs	164
General	164
FraudOne Service Programs requirements	164
Overview FraudOne Service Programs	165
Service Programs use and functionality	166
Standard Service Programs GUI.....	167
Log Files.....	168
Viewing the progress of files.....	169
Re-starting a process.....	170
Using the Data Files.....	170
Using the Data Viewer.....	171
Using Report Files with the Service Programs.....	171
Service Programs configuration	172
Service Programs	172
Account Loader	172
Account Loader process.....	172
Installing the Account Loader.....	173
Starting the Account Loader.....	173
Limitations of the Account Loader.....	174
Image Loader	174
Image Loader process.....	175
Installing the Image Loader.....	175
Starting the Image Loader.....	176
Limitations of the Image Loader.....	176
Signature Reference Filter	177
Signature Reference Filter process.....	177
Installing the Signature Reference Filter.....	178
Starting the Signature Reference Filter.....	178
Limitations of the Signature Reference Filter.....	179
SignCheck Getter	179
SignCheck Getter process.....	180
Installing the SC Getter.....	180
Starting the SC Getter.....	181
Limitations of the SC Getter.....	181
SignCheck Putter	182
Use of Primanota processing.....	182
Installing the SC Putter.....	182
Starting the SC Putter.....	182
Limitations of the SC Putter.....	183
Day's Final Processing	183
Installing the Day's Final Processing.....	184
Starting the Day's Final Processing.....	184
Limitations of the Day's Final Processing.....	185
Fraud Feedback File Loader	185
F3-Loader Process.....	185
Installing the Fraud Feedback File Loader.....	186
Starting the Fraud Feedback File Loader.....	186
Limitations of the Fraud Feedback File Loader.....	187
XML-Loader	187
XML-Loader process.....	188
Limitations of the XML-Loader.....	188

Configuration of the Service Programs	188
The service.properties configuration file	188
Report Files	217
Account Loader Report File.....	217
Image Loader Report File.....	217
Signature Reference Filter Report File format.....	219
Fraud Feedback File Loader Report File format.....	220
Trace levels	221
Work File processing	222
Requirements for use of Work Files.....	222
Reject File processing	222
Advantages of Reject Files.....	223
Password encryption	223
Chapter 13 Archive Interface Server	223
Overview Archive Interface Server	223
Archive Interface Server installation	224
Database components	224
DB2	224
Oracle	224
MS SQL Server.....	225
Java Client components	225
Server components	225
Starting the Archive Interface Server	225
Starting from the Server Manager.....	225
Starting from the Command Line.....	225
Archive Interface Server configuration	226
custom.properties	226
sizes.properties	227
archive.properties	227
ams.properties	230
asdatabase.properties	230
amsdatabase.properties	230
Status Entries	231
Chapter 14 Appendix	233
Property files overview	233
Business Model property files	233
Thin Client property files	236
Archive Interface Server property files	237
Parameter overview	237
Server Manager parameters	237
Administration Client parameters	257
Java Client parameters	258
Thin Client parameters	261
Thin Client servlet parameters	282
Engines parameters	282
Archive Interface Server parameters	291
Port number defaults	299
FraudOne Server messages	300
Messages for SignBase Servers	300
Messages for SignCheck Servers	301
Messages for ASV and AFD Servers	302

Messages for STV Servers	302
GIA configuration for PreProcessingEngine	302
Purpose	302
Possible feature overview	303
Configuration of the PPE features	304
Parameters for all features.....	304
FeatureID 700 - Cropping signatures.....	304
FeatureID 730 - Cleaning the cropped signature.....	304
FeatureID 740 - Searching signatures.....	305
Example of a GIA PPE configuration	305
GIA configuration	306
Purpose	306
GIA configuration for Statistical Analysis Engine (SAE)	307
Purpose	307
Possible feature overview	307
Configuration of the SAE features	309
Parameters for all features.....	309
FeatureID 1793 - Serial number verification in issued serial number ranges.....	309
FeatureID 1794 - Serial number verification in observed serial number Ranges.....	310
FeatureID 1795 - Serial number duplicate verification.....	310
FeatureID 1796 - Amount range verification.....	310
FeatureID 1797 - Checkv velocity verification.....	311
FeatureID 1798 - Check amount highlighting.....	311
FeatureID 1799 - Account number highlighting.....	311
FeatureID 1800 - Branch/transit code highlighting.....	311
FeatureID 1802 - Serial number verification in observed serial number ranges for external accounts.....	312
FeatureID 1803 - Serial number duplicate verification for external accounts.....	312
FeatureID 1804 - Amount range verification for external accounts.....	312
FeatureID 1805 - Check velocity verification for external accounts.....	312
FeatureID 1806 - Check amount highlighting for external accounts.....	313
FeatureID 1807 - Account number highlighting for external accounts.....	313
FeatureID 1808 - Branch/transit code highlighting for external accounts.....	313
Example of a GIA SAE configuration	314
ARV configuration example	314

Preface

The *Kofax FraudOne Administrator's Guide* serves as a reference manual for the installation, use, and configuration of the FraudOne system. This document covers the FraudOne concepts, deployment, routine maintenance, configuration and troubleshooting procedures.

This manual is organized into the following distinct but complementary parts:

Introduction

Provides an overview and system architecture of the FraudOne system.

[Preface](#)

[FraudOne architecture](#)

Data base

Describes the usage of supported database products and their tasks in conjunction with the FraudOne system.

[FraudOne databases](#)

[FraudOne Server Manager](#)

Clients

Describes installation, tasks and areas of use for the Client programs.

[FraudOne Administration Client](#)

[FraudOne clients](#)

[SignCheck Engines](#)

Process control

Describes the modules to optimize the document check and to manage the technical workflow and system supervision.

[SignCheck CRS](#)

[Kofax FraudOne Reports](#)

[Server Monitor](#)

[FraudOne SNMP](#)

Service programs

Describes the modules which control the data flow between external sources and the FraudOne System.

[FraudOne Service Programs](#)

[Archive Interface Server](#)

References

Contain a collection of additional information provided by Kofax.

[Related documentation](#)

It is assumed that FraudOne administrators have thorough understanding of databases, the current network layout, equipment, and limitations. This requires insight into the internal processes and technical integration, and a detailed experience in maintaining ASCII-based configuration files (INI files, Java properties and ZIP archives).

Kofax highly recommends using the tools and components provided in the FraudOne installation documentation. Consult your Kofax technical representatives before using any tools or procedures that are not described in this manual.

Related documentation

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

https://docshield.kofax.com/Portal/Products/en_US/FO/4.4.2-c5l5th79bw/FO.htm

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

Guides

- *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 Java Client Customization Layer*
- *Kofax FraudOne License Management*
- *Kofax FraudOne Report Component Installation Guide*
- *Kofax FraudOne Service Program Configuration*
- *Kofax FraudOne SignCheck Result Codes*
- *Kofax FraudOne Standard Reporting Features and Statistics*
- *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 Common API Specifications for GIA Engines*
- *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*

Training

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

Get help for 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.

FraudOne architecture

This chapter focuses on the FraudOne system architecture and its 'out-of-the-box' standard functionality. The system described in this chapter as well as the configuration settings may vary depending on the products you have purchased.

Overview FraudOne architecture

FraudOne is a flexible product suite that is built around the customer's signature verification needs. The complete solution provides several scalable components that can be used in order to:

- Create and maintain a customer reference database
- Create and maintain a signature database
- Use the database information for signature verification
- Perform visual verification of signatures cropped from payment forms and checks
- Use the FraudOne Engines for the use of automatic signature verification as well as for automatic payment image analysis, e.g. check stock verification
- Use the visual Client interfaces to access reference images and information
- Perform remote system administration
- Perform online verification of signatures collected from pressure sensitive pads and Tablet PCs

In their simplest configuration, the FraudOne components provide a solution for storing data and images in a database for the purposes of visual signature verification. The system architecture can be scaled to encompass a complex solution for fully automatic verification of signatures and signing rules in real-time.

The core products of the FraudOne are:

- **SignBase:** A solution for storing and retrieving signature information related to bank accounts
- **SignInfo:** A solution for storing and retrieving image information (archive images, account cards, or any other document images) related to bank accounts
- **SignCheck:** A framework for the clearing of check images using both visual and automatic verification.

The FraudOne components provided by the above products include the following components:

- File based interfaces that are used to create and manage customer information
- Database components that are used for the storing of reference and processing data
- Visual Clients that are used within the FraudOne system
- Monitoring and Administration tool

This chapter describes the FraudOne 3-tier client-server architecture which enables the communication between the FraudOne databases, the FraudOne Clients, and the bank's IT infrastructure. The architecture is comprised of the following three tiers:

1. FraudOne database layer consisting of the SignBase and SignCheck databases. This includes all file-based auxiliary programs also known as Service Programs for interfacing to the bank's back-end systems. Through the exchange of file-based data, the Service programs enable the integration of the FraudOne system into the IT infrastructure.
2. Application server layer which provides the communication between the FraudOne servers and the FraudOne Visual Clients and Engines
3. Visual Clients that are used for access, maintenance, and control of the FraudOne databases

The following diagram describes the FraudOne system in its standard functionality:

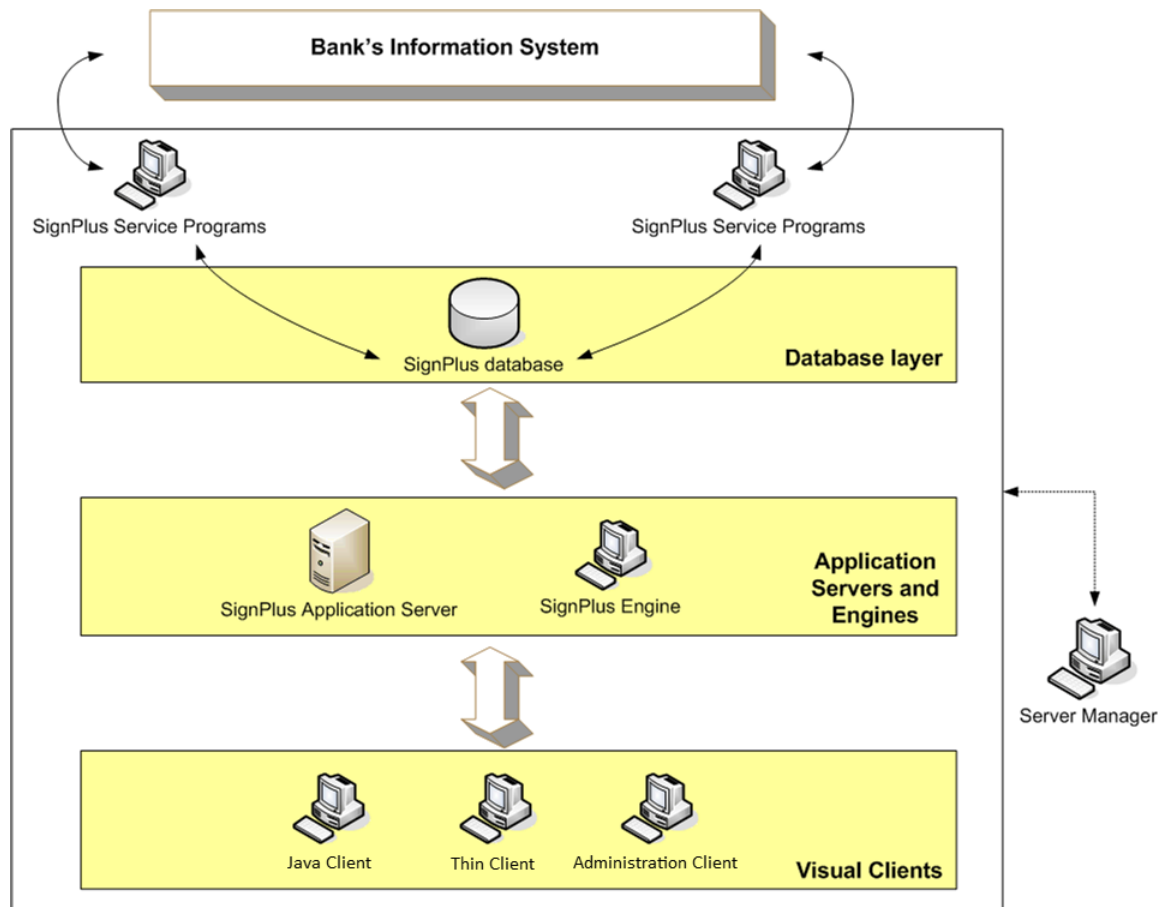


Figure 1: FraudOne system in standard functionality

The FraudOne Server Manager controls the Application Servers and Engines as well as the FraudOne Service Programs.

Server Manager

The FraudOne Server Manager is the single control interface of the FraudOne components (except the Thin Client) in the three tier architecture.

The Server Manager can be used to start and stop selected FraudOne components at predefined times. It also provides a calendar scheduler for day-dependent scheduling of components.

The operation of the Server Manager and the programs it manages is controlled by entries in the central configuration database. These entries are edited using the Administration Client and consist of topographic elements that govern the machines where the Server Manager is used and the types and numbers of the programs that the Server Manager can start on each machine. See the [FraudOne Server Manager](#) chapter for a further description of the Server Manager functionalities and configurations.

Attach Manager

The Attach Manager program acts as an intermediary and allows for an efficient management of Visual and non-Visual Client requests to the database and servers. It receives all the messages from the Clients and distributes them to the available application servers.

FraudOne interfaces

The FraudOne standard functionality is designed to work in conjunction with a bank's existing information systems. The FraudOne components provide file interface possibilities, which enable the migration of existing customer information into the required FraudOne formats.

The bank's IT infrastructure usually contains the following two information sources:

- Customer Information System (CIS) which includes the customer and account information
- Payment Information System (PIS) which includes the check data and check images

To integrate into the bank's IT infrastructure, data needs to be migrated from the CIS in order to build the reference data information for FraudOne. The information is collected by the bank, converted into a pre-defined FraudOne format, and finally loaded into the FraudOne system via a file interface. The data includes account information, customer information, reference signatures (as images), and signing rules. If check stock verification is used, then reference check stock images may also be loaded.

In order to prepare the system for a check processing production cycle, the information needs to be loaded from the PIS into FraudOne. The SignCheck Getter is used for migration of data from the PIS into the SignCheck database. The Getter interface reads data files consisting of image and data for the checks that need to be verified.

Once the daily processing has been completed, FraudOne will present the results to the bank's back-end systems (usually the PIS system again) in the form of a results file. This file is called the "Putter file" and is named after the Service Program (the "Putter") that is used for exporting the results after processing has completed. The results file is then used by the PIS to make the "Pay" or "No Pay" decisions on individual check items.

At the end of the SignCheck production cycle the SignCheck Day's Final Processing (DFP) program performs the cleaning of the SignCheck database and prepares it for the next processing cycle.

The interfaces described above are further described in the [FraudOne Service Programs](#) chapter.

The diagram below displays the interfaces between the Banking Information System and the FraudOne databases via the use of the FraudOne Service Programs:

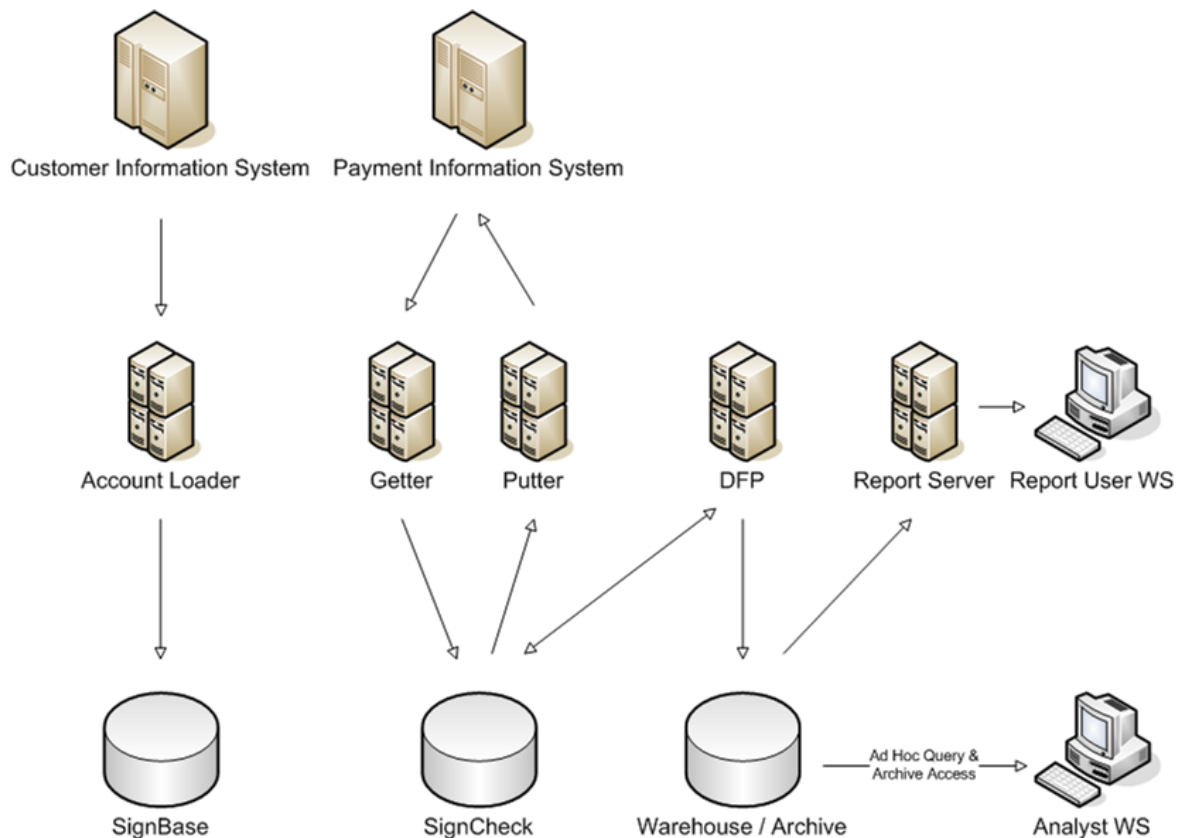


Figure 2: FraudOne Service Programs interfacing with bank's information system

FraudOne databases

The FraudOne databases are used for storing customer reference information (SignBase), check data and images for signature and check stock processing (SignCheck) and archive and statistical information (Warehouse and Archive).

The data is stored in a database management system. The supported platforms include DB2, ORACLE and MS SQL Server. The data for both SignCheck and SignBase may be located on a single database management system. Alternatively, it is possible to distribute the data such that the SignBase and SignCheck data reside in separate physical databases. The Warehouse and Archive data should always reside on a separate physical database to protect the production databases from adverse performance impacts due to ad-hoc queries.

SignBase

SignBase enables the creation and maintenance of reference data used for daily processing tasks. The following are examples of the type of information stored in the reference database:

- Customer information (e.g. customer number, name, valued customer, etc.)
- Account (e.g. account number, name, etc.)

- Signatory (name, validity dates, etc.)
- Signature images
- Signing Rules
- Document images and data (SignInfo only)
- Check Stock images

The server-side components of the SignBase main module cover database access and data consistency as well as load-balancing of the client side requests via an Attach Manager. The information that is stored in the SignBase database can be accessed by a user using the client-side components (Java Client and Thin Client). These Visual Clients are used for data and image presentation. Since SignBase is the main repository for reference information, all other FraudOne components rely on SignBase for processing. So if a SignCheck verification of check images for a particular account is required then the account information will be retrieved out of SignBase by the appropriate components.

SignInfo

SignInfo is used to store images related to an account within SignBase. Although, from a product perspective it is separate from SignBase, the SignInfo information physically resides within the SignBase database. A customer may choose to store signature information (SignBase), signature and image information (SignBase and SignInfo combined), or only image information (SignInfo only).

SignInfo can be used to store the following information as reference data:

- Images of account opening forms
- Data from signature cards
- Individual signature images (not for Automatic Signature Verification ASV)
- Identification photos of account holders
- Scanned identity documents
- Any other Account information images such as restriction documents

In this manner SignInfo can be used as an add-on to the regular SignBase features for presenting non-signature documents to the bank operators.

SignCheck

The SignCheck module is used for the automated verification of signatures and signing rules, and/or check stock images against reference data that is stored in the SignBase database.

SignCheck employs a separate database tables for the data that it requires during a day's processing cycle. This is because the data will change daily according to the in-flow of items to be processed.

SignCheck provides the ability to load cheque data and images into the SignCheck database for processing. Once loaded, the items (images and data) will be processed as part of a verification workflow. The SignCheck main module with its built-in SignCheck Workflow provides a central framework for complex item review operations. The exact flow of the items during the verification process will vary from installation to installation. However, in general the workflow will include some combination of both automatic verification (e.g. ASV) and visual verification (VSV) queues into which items will be routed during the verification cycle.

With the use of the FraudOne Engines (e.g. Automatic Signature Verification, or Check Stock Verification) a variety of automatic verification processes can be implemented into the workflow. The workflow can be configured such that a review operator can exclusively focus on suspicious items only (i.e. that have been previously rejected by the automatic signature and rules verification). The database access and load balancing of SignCheck components are provided by the Attach Manager.

Data Warehouse and Archive

This separate database is used to store statistical information and archived check data and images as well as the results of the processing of those cheques. Access to the statistical information is provided through predefined reports via the Report Server as well as through ad-hoc queries. The archived check data is accessed using the SignCheck Thin Client.

Application Server layer

The FraudOne Application Server layer contains all server-side components of the FraudOne architecture. It is responsible for providing the infrastructure for these server components to function properly. Additionally, the application server layer enables the communication between the FraudOne databases and the FraudOne Clients.

The application server layer includes the following server-side components:

- SignBase application server
- SignCheck application server
- FraudOne Engine application servers
- SignCheck Workflow
- Attach Manager which is used for load balancing

The following diagram depicts the communication between the FraudOne Application Server and the SignBase and SignCheck databases. The Workflow Router is used for moving the check items from one processing step to the next:

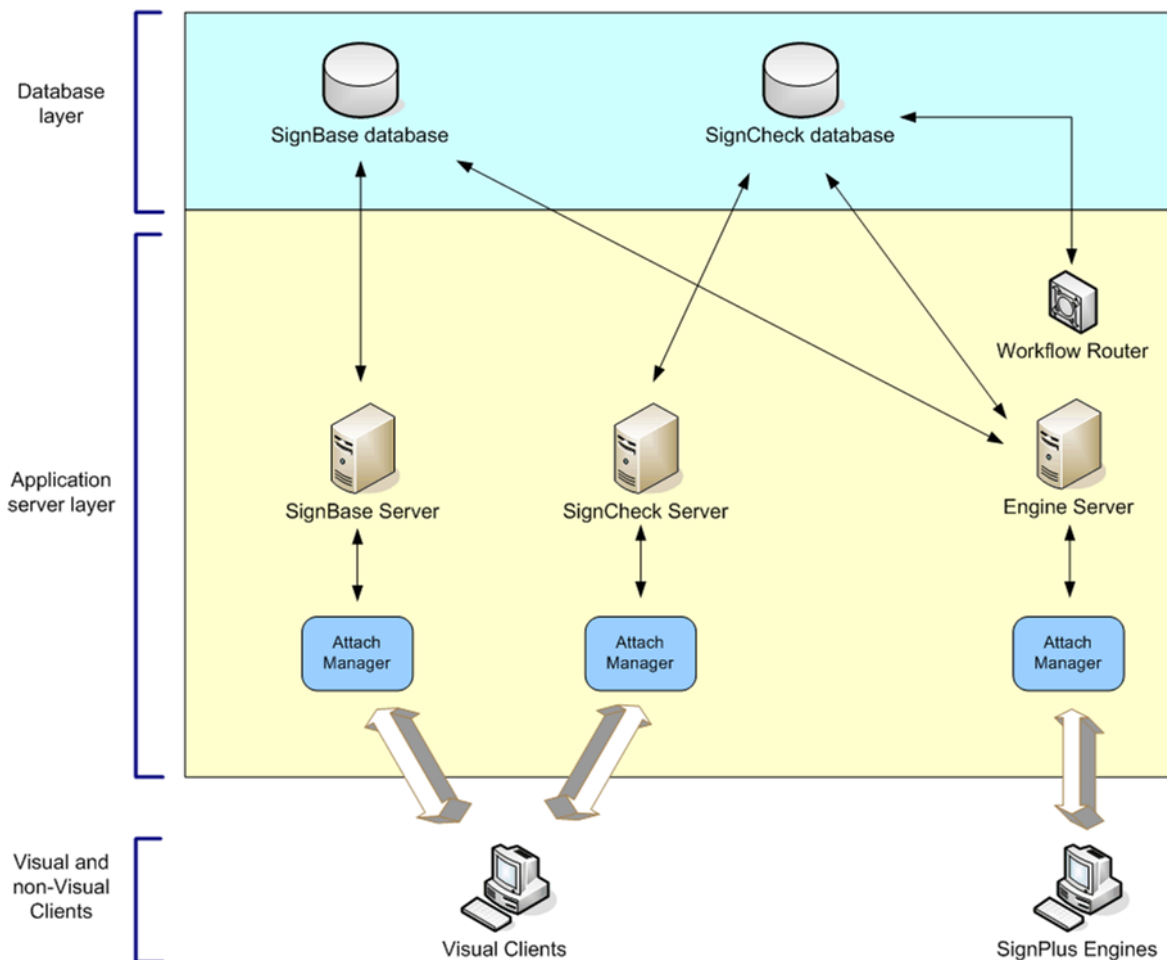


Figure 3: FraudOne Application Server and Workflow Router

SignBase Server

The SignBase server enables database access for SignBase requests, i.e. requests for reference data that resides within the SignBase database. In addition to providing the connectivity, it also ensures data consistency when modifications are performed.

The FraudOne system may be configured to use several SignBase server instances depending on the load that needs to be processed. The load balancing of the requests is performed via the Attach Manager which manages the incoming requests and decides which of the SignBase servers should service it.

SignCheck Server

The SignCheck server enables database requests for SignCheck requests, i.e. requests for data that resides within the SignCheck database. The FraudOne system may be configured to use several SignCheck server instances depending on the load that needs to be processed. The load balancing of the requests is performed via the Attach Manager which manages the incoming requests and decides which of the SignBase servers should service it.

FraudOne Engine Server

When performing automatic signature verification, or check stock verification, the use of a FraudOne Engine is required. The Engine retrieves the account information on a check and compares it against the check information and check image in the SignCheck database as well as the reference data contained in the SignBase database. The FraudOne system can comprise of the following available Engines:

- ASV: Automatic Signature Verification
- ARV: Automatic Rule Verification
- APSV: Automatic Payment Signature Verification
- GIA: Generic Image Analysis

The [SignCheck Engines](#) chapter further describes these components as well as the configuration possibilities.

SignCheck Workflow Router

The SignCheck Workflow Router is an independent server process that routes the check items from one processing step to the next. The customer specific Workflow Router can be configured to include the use of automatic verification followed by one or more visual verification steps.

The following describes the Workflow Router's functions in the FraudOne system:

1. The FraudOne Engine sends a request (via the FraudOne Engine server) for the next available item for processing.
2. The FraudOne Workflow Router receives this request and searches the SignCheck database for the next item that is in a "ready" state (ready for processing).
3. The Workflow Router locates this "ready" item and sends an item identifier back to the FraudOne Engine server. This item identifier contains the location of the item in the SignCheck tables. This information is used by the Engine server in order to retrieve the "ready" item from the SignCheck database.
4. With this item identifier the FraudOne Engine retrieves the next available item for processing.
5. Once the FraudOne Engine has finished the verification process a result message is written into the SignCheck database tables.

Refer to the [SignCheck Engines](#) chapter for more details on functionality and configuration of this component.

FraudOne visual clients

The FraudOne Clients allow for a visual interface to be used in the FraudOne environment. The Java and Thin Client provide the user with a means to view and to perform the general maintenance of customers and accounts.

The FraudOne Clients can be integrated into existing front office information system by means of the FraudOne Teller interface. It allows users to query info from the SignBase database using the Teller interface.

Java Client

The Java Client provides Graphical User Interface functionality for the use of physical and automatic verification of checks, images, and signatories. The Java Client accesses the SignBase database in order to perform maintenance on the reference data. It can also access the SignCheck database in order to perform visual verification of check images and signatures as well as perform maintenance on reference data.

Thin Client

The FraudOne Thin Client is a web application which runs on a web server within the IT infrastructure. It is accessed via supported web browser such as Internet Explorer, Mozilla, or Firefox. The Thin Client enables access to FraudOne functionality through a company's network using standard Intranet and Internet technologies. The Thin Client accesses the SignBase database for the display of reference data. It can also access the SignCheck database in order to perform visual verification of check images and signatures.

The diagram below describes the Visual Client communication requests with the Application Servers:

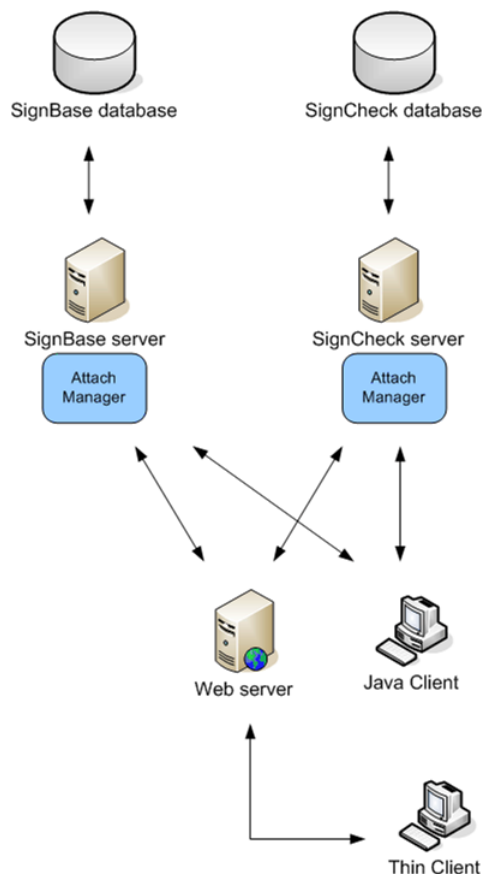


Figure 4: FraudOne Clients communication

Administration Client

Administration Client is a standalone Java application that provides graphical user interface for system-level configuration of FraudOne.

User Administration module allows for maintaining users and managing their access rights.

System Configuration provides access to system components' configuration.

Topography Editor manages the FraudOne application server layer and service program processes that are executed on each configured FraudOne server system.

Blacklist Administration contains means for maintaining blacklist.

CRS Editor is a visual tool for working with rule graphs that determine CRS workflow.

FraudOne concepts

The following describes the common concepts used in the FraudOne system.

FraudOne Data Model

The FraudOne Data Model comprises a collection of objects for modelling the data and the relation between objects in the system. An object can be referred to as a signer's individual signature, data such as name, role, signing rules and rules for specific accounts or customers.

FraudOne may be used in two modes:

1. **Customer Model:** In the Customer Model, a customer may be organized in several accounts. This means that there is a "one-to-many" relationship between a customer and several accounts and a "one-to-many" relationship between a customer and several signatories. The data model is organized in a tree structure where the top-level (or root) object is the Customer object.
2. **Account Model:** In the Account Model, the customer object is considered equivalent to the account object. Therefore on a technical level, there is a one-to-one relationship between a single customer and a single account. The customer object however is not visible in any of the front-end applications. The data model is organized in a tree structure where the top-level (or root) object is the Account object. Regardless of which model is used, within the database, the customer object is still the top level object that contains the accounts.

The data model tree below depicts the FraudOne Customer Model:



Figure 5: FraudOne Data Model

Where the Customer Model contains the combination of the following models:

- Account: contains the account information
- Signatory: contains the text information of the person that is allowed to sign
- Rules: contains signatory rule information
- Signatures: contains the signature images of the customer and can be defined as
 - SIGNATURE_M (monochrome signature)
 - SIGNATURE_G (grayscale signature)
 - SIGNATURE_D (dynamic signature derived from a signature pad)
- Groupin: contains the signature information of all persons included in the group
- Rule Group: contains the relationships for complex rules
- Stock Image: contains the customer check stock image and information

All models use the following primary key identifiers:

1. Bank Number (BNO)
2. Country ID
3. Bank Code (Branch Number)
4. Customer Number

Historisation

When querying the database, a user will generally retrieve the information for the current date. However, FraudOne also provides the ability to query the database to retrieve information as it was at an earlier point in time. The process of saving this historical information is called Historisation.

Historisation enables the user to query customer or accounts in the SignBase database for a particular date. If a certain customer, account or signatory data is changed or deleted in SignBase, the initial values are not removed from the database. This important feature also enables the user to identify which signatories and signing rules were valid for a particular item on the date that the item was cleared. All delete actions in SignBase result in logical deletions only, so that the data may be historised rather than physically deleted from the database. Physical deletion is only possible with the Account Loader Service Program.

Variants

The signature or check stock reference images are stored in the database for a particular account. Over time, the need may arise to allow variations on the original reference images. For example, if a person's signature changes over time or if a new check stock has been introduced. Another application may be to enable the capture an up-to-date, high-resolution and high-quality signature during the processing of checks. For this purpose, FraudOne has introduced the concept of Variants. Variants represent a valid additional image that can be stored alongside the original reference data as a variation.

Variants are generally used in the following three scenarios:

- When a signatory is created, the quality of the initial signature might not be good enough for Automatic Signature Verification (a FraudOne Engine component), and therefore an additional variant is stored in the reference database.
- People's signatures might tend to change over time.
- A person may sign differently when signing for different accounts (using abbreviated first names, omitting or adding titles).
- To build the reference database from scratch.

There are two types of variants supported by FraudOne. Both of these types apply to both Signatures and check stock images:

- **Assigned variants:** Variants can be assigned to one or two authorized signatories. The valid signing rules for the signatory also apply to the assigned variants. If a variant refers to two authorized signatories with different signing rules, the higher amount limit would apply. Only assigned variants can be used for Automatic Signature Verification or check stock verification
- **Unassigned variants:** Unassigned variants are attached to a customer, but not yet to a specific authorized signatory of the given customer or account. They are usually assigned to an existing signatory at a later stage, either manually or in some cases even automatically. Unassigned variants have no effect on Automatic Signature Verification (ASV) or check stock verification.

Four Eyes Principle

It is common practice that any changes in a customer or account have to be approved by a bank employee other than the one who has made the changes. This system of verification and approval by a second person is called the Four Eyes Principle. Besides the verification of all changes,

FraudOne can be configured to validate the data changes only for a certain percentage of the changes. The defined values can vary between the different customer types (private, corporate or other) enabling a customer specific balance between speed, productivity, and security.

Authorization models

FraudOne provides two authorization models, Active Directory integrated model and FraudOne native model.

FraudOne native authorization is the only model available before Release 4.4.2 and uses the SignBase database to store user and group objects and their authentication (passwords etc.) and authorization attributes. Using this model, a FraudOne administrator using the FraudOne Admin Client must define all FraudOne users and FraudOne user groups and their associated group membership structures and their access rights to the various FraudOne features.

Active Directory integrated authorization places all user and group management associated with authentication under the control of the customer's Active Directory installation and its administrators. The FraudOne administrator using the Admin Client is only responsible for the assignment of access rights for the various FraudOne features to existing Active Directory users and groups.

When Active Directory integrated authorization is active, none of the following FraudOne functions can be used – these functions are all carried out only by Active Directory or the Active Directory administrators:

- Creating, changing or deleting user objects.
- Creating, changing or deleting group objects.
- Group Membership management.
- Password expiry, history and complexity checking.
- User lockout and lockout reset on authentication failure.

Assigning FraudOne feature access rights is done by using the FraudOne Admin Client to:

- Import an existing Active Directory User or Group and
- Assign or remove the required FraudOne access rights to these Users and Groups.

The recommended approach here is to assign access rights ONLY to Groups and to use Active Directory group membership to control the authorization to the FraudOne features. When used in this way, FraudOne administration never has to be concerned with individual users from Active Directory and this significantly reduces the administrative workload. All users who are a member of at least one Active Directory group that has been granted any FraudOne access right will be able to log on to a FraudOne client even if their user identifier has not been imported into FraudOne.

The only way to switch between the Active Directory integrated and FraudOne native authorization models is by using the `SBDTab.cmd -resetusermanagement` option. Notice that the Admin Client also needs to be configured to enable Active Directory administration.

Best practice for configuring FraudOne's Active Directory integrations is:

- The authenticating SignBase server process should be run on a machine in the Active Directory domain.
- The authenticating SignBase server process should be run under a domain user with read only access rights to the Active Directory catalog information for the User, Group and Group Membership information.

With these restrictions the server needs no configuration parameters to set up its access to Active Directory, the installation security is maximized since no technical user information is needed and

the best possible system performance is achieved since the server does not need to continually reauthenticate with Active Directory with a different user-id.

Multiple bank support

Clearing houses, service providers, processing houses and large banks may provide verification services for several different banks while implementing only a single FraudOne installation. Therefore, FraudOne supports multiple banks per installation. Users may easily switch between banks by using the "Select BNO" command, which is available in the file menu of the FraudOne Java and Thin Client. Access to BNOs can be restricted depending on the client sites and user access.

Multiple Bank support is guaranteed by provision of a BNO field for each data record. This means that data of different banks can be segregated by a three digit BNO value.

Note Alphanumeric BNO's are not supported.

The valid range for BNO numbers starts from 001 to 998, while the value of 999 is reserved for internal use only.

FraudOne databases

General

This chapter describes the FraudOne database installation requirements and procedures. The availability and limitations of certain programs and functionalities in your installation may vary depending on the products you have purchased.

FraudOne can store also historical and statistical information for analysis by the customer. This [Data Warehouse](#) information and the database structures used to contain it are described separately below.

Overview FraudOne databases

The FraudOne databases are used for storing customer reference information (SignBase) as well as storing check data and images for signature and check stock processing (SignCheck). The data for both SignCheck and SignBase may be located on a single database management system. Alternatively, it is possible to distribute the data such that the SignBase and SignCheck data reside in separate physical databases. The following describes the different databases:

- **SignBase:** Enables the creation and maintenance of reference data used for daily processing tasks. The following are examples of the type of information stored in the reference database:
 - Customer information (e.g. Customer no, name, valued customer, etc.)
 - Account (e.g. Account number, name, etc.)
 - Signatory (name, validity dates, etc.)
 - Signature images
 - Signing Rules
 - Document images and data (SignInfo only)
 - Check Stock images

- User access information
- Configuration data for all FraudOne components
- **SignCheck:** Stores the check data and check images that are used in the daily processing cycle. It also stores the processing results and the SignCheck Workflow status information.

This chapter describes the SignBase and SignCheck databases as well as installation procedures of the supported database management systems DB2, ORACLE and Microsoft SQL Server. In general, the versions of the database management systems that are supported by their respective manufacturers are also supported for use with FraudOne. A separate document *FraudOne Technical Specifications* provides details of the actual supported versions.

The initial installation of the databases requires the allocation of physical disk space to contain the SignBase and/or SignCheck database tables and construction of a suitable authorization scheme to allow access to the database contents only to authorized people and programs. Consult your Kofax business representative for the accurate spacing requirements.

The use of the Validation program, which performs a consistency check and correction on the SignBase and SignCheck databases, is described in [SignBase and SignCheck database Validation](#).

Prerequisites and consideration

The following requirements must be considered before a database installation procedure:

- Only one type of database management system should be installed for the FraudOne databases.
- For the SignBase database a high reserve capacity is necessary for the storing of permanent data (especially for historised data and variants).
- For the SignCheck database a high reserve capacity is necessary for the storing of temporary data and documents
- If several databases are in different locations then remote database nodes should be configured based on the network protocol of the product
- Additional programs need to be installed if connecting to the database via a mainframe computer. Refer to the vendor's installation guide for more information.
- The Database management systems (DB2, Oracle and Microsoft SQL Server) require a client installation including an ODBC driver on the computer where an Application Server is running. Only the ODBC drivers of a suitable release level supplied directly by the corresponding database manufacturer are supported. If a computer is running a FraudOne Service Program a client installation is necessary unless a JDBC Type 4 driver is used.

SignBase database

The SignBase database tables consist of customer information (CUSTOMER), account information (ACCOUNT), authorized signatories (SIGNATORY), signature powers (e.g. RULE, RULE_GROUP, GROUPIN) and reference signatures. The signature images of the authorized signatories are kept in separate tables for the different processing steps such as recording, comparison, and forgery detection.

The FraudOne user tables (SPX_USER, SPX_GROUP, SPX_UR, SPX_UG, SPX_GR) and the rules tables (SPX_RIGHTS) are also stored in the SignBase database. The type of fields used in the standard tables depends on the customer's system requirements and the purpose (e.g. mainly visual recall of signatures vs. pure automatic document processing in payment transactions).

Configuration management data for engines and workflow components can be stored also in the SignBase database in the table address map table (SP_ADDRESSMAP) and the configuration element table (SP_CONFELEM).

SignCheck database

The SignCheck database contains the check data and check images, information used for SignCheck workflow control, and the processing cycle statistics.

The SignCheck tables consist of information that must be transferred from document-based payment transactions to the system in order to conduct effective automatic signature verification.

During the processing by the FraudOne Engines (ASV, VSV, GIA), the SignCheck tables contain the written or printed information from the documents, such as account number, amount, etc. (SC_INTERFACE table). The status variables for each document running through the individual steps in the workflow (SC_WORKFLOW table) are as much a component of SignCheck as the front and back of the document images and the signature snippets in their various quality levels (SC_IMAGE table).

Creating and maintaining the databases

The following describes the database installation requirements as well as the installation procedures using the following database management systems:

- IBM DB2
- Oracle
- Microsoft SQL Server

Refer to the document *FraudOne Technical Specifications* for the required versions for each database product. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Important Knowledge of the database management system product as well as specific database systems is required before performing the following installation and maintenance procedures.

Kofax provides a variety of interfaces to be used with the databases. Unless you have explicit consent from Kofax, do not make any changes to the database and database contents with tools and methods other than the ones specified by Kofax.

General information for all databases

Database administration

Administration of the database(s) is the responsibility of customer database administrators (DBA) unless otherwise specified in a customer contract with Kofax. The following information is intended as a guide for the DBA and is only a suggestion. Specific customer requirements for performance and / or security may mean that other database configuration options are necessary.

FraudOne will adapt automatically to (or in some cases need to be configured to comply with) any DBA configuration choices. In most cases, FraudOne can treat all of the supported database

management systems identically – exceptions are noted in the sections for the specific database management system below.

Schema names

FraudOne allows the schema name used for all database objects created by the FraudOne table creation scripts to be chosen freely by the DBA. The chosen name must be entered into the respective configurations of the FraudOne components that access the database and in the properties files for the table creation programs.

Codepages and national language characters

FraudOne is largely independent of the code page selected for the database. The FraudOne programs all operate internally using Unicode encoding and are independent of the customer's national language choices. When choosing a database code page, the DBA must ensure that the code page chosen can contain all characters needed for the national language(s) that are used. FraudOne will use Unicode data fields in the database if these are available with the DBA's choice of code page.

National language data is stored in the database only for fields where this constitutes a normal use of the data (names, free text fields and descriptions). Key fields and encoded data (for example, account numbers, dates, data type codes and Kofax internal use only data items) may not contain national language characters and use standard ASCII database fields where these are available.

Allocation of disk space

All database management systems provide the DBA with abstractions that allow the flexible provisioning of disk space to contain table data and indexes. Terms used for this are "table spaces", "file groups" or "storage groups".

The DBA is free to choose a space allocation model for the FraudOne tables that matches their standards and their performance criteria. This can range from placing all data into a single "table space" to assigning a separate table space for each table, for example.

The FraudOne table creation programs (SBDebTab and SCDebTab) use configuration files to allow the DBA to configure the programs in order to place the generated tables and indexes into the correct disk space that has been configured for them.

Notice that these properties files are an exception to the central database storage of configuration data for the FraudOne product. This is required since the database does not exist at the time the SBDebTab and SCDebTab programs are used.

This configuration is intended to cover most of the normal cases found in practice so that the generated DDL from the programs can be used directly as generated, without editing. For extreme cases, the output DDL can be routed into files where it can be edited and then executed. Such editing should be restricted to the mapping of tables and indexes to specific disk storage locations. Other changes to the data structure defined in the DDL are not allowed and may lead to incorrect operation of the product.

Authorization

FraudOne requires only one "technical" database user for production operation. This user needs only rights to modify the data held in all FraudOne database tables (and to execute any FraudOne stored procedures where the installation uses these). It is recommended that the authorizations

required be assigned to a group or role and then the individual users needing these rights be authorized to the role or added to the group as required.

The DBA is free to extend this authorization scheme to match the local standards and practices.

The FraudOne table creation programs (SBDebTab and SCDebTab) can be configured to grant the required rights to the technical user (or role or group) automatically during the table creation process.

The user running the table creation programs requires rights to:

- Use (allocate objects in) the table spaces
- Create tables, views and indexes in the schema used
- Create stored procedures
- Grant rights to modify the data in the created tables and execute the created stored procedures

Trusted connection

The normal configuration for the FraudOne access servers includes one or more "technical" users and their passwords used to connect the access servers to the database(s). While this authentication data can be encrypted to prevent the data being stolen from the configuration files and misused by unauthorized people to access the database, a more secure option is to use a "trusted connection" to the data base. This uses the Windows authentication of the user running the FraudOne access server programs (or the service where these programs are started) directly to connect to the database. In this case, no database users or passwords need to appear in the configuration files.

FraudOne fully supports this authentication model in so far as the database management system allows this and can be configured to use it securely.

Table creation programs

The programs SBDebTab (for SignBase) and SCDebTab (for SignCheck) are provided to create the tables needed by the FraudOne programs and their associated supporting structures. The Data Warehouse tables have their own scripts described elsewhere.

The configuration of these programs is done using a properties file for each program (so, SBDebTab.properties and SCDebTab.properties) and the use of these properties files and the program options used to invoke the programs are described here.

SignBase table names

The **SBDebTab** program manages the following tables:

```
CUSTOMER
ACCOUNT
SIGNATORY
SIGNATURE_M
SIGNATURE_G
SIGNATURE_D
RULE
RULE_GROUP
GROUPIN
STOCKIMAGE
```


NCSTOCKIMAGE
 ACCOUNT_IMAGE
 DOCUMENT
 MASK
 SP_ADDRESSMAP
 SP_CONFELEM
 SPX_USER
 SPX_GROUP
 SPX_RIGHT
 SPX_UR
 SPX_GR
 SPX_UG
 PROTOCOL
 SB_STATISTICS
 REFERENCESTATISTICS
 SP_CFMI

SignCheck table names

The **SCDebTab** program manages the following tables:

SC_INTERFACE
 SC_IMAGE
 SC_RESULT
 SC_WORKFLOW
 SC_PRIMANOTA
 SC_STATISTICS
 SC_CRSRULENAMES
 SC_PROTOCOL

Command options

The available command options are listed in the following table:

Option	Description
-?	Displays command line help (and ignores any other options specified).
-d <db name>	The name of the database where the tables are to be created.
-u <user-id> -x <password>	Authentication parameters for the DBA user creating the tables.
-s <db server> -p <port> -p dynamic	The port and physical (network) name or address of the server where the database can be found (not required if the database name alone can identify the required connection). Dynamic is used to indicate the use of Microsoft SQL Server dynamic port assignment feature.
--indexonly	Instructs the program to delete and recreate the standard index set (for the selected tables) without changing any other database structures.

Option	Description
<code>--storedprocedureonly</code>	Instructs the program to delete and recreate the stored procedures (if any) without changing any other database structures.
<code>--triggeronly</code>	Instructs the program to delete and recreate the table triggers (if any) without changing any other database structures.
<code>--primarykeyonly</code>	Instructs the program to delete and recreate the table primary keys (for the selected tables) without changing any other database structures.
<code>--resetoidonly</code>	Ensures that the database management system assigned counters used to allocate new database object identifiers (OIDs) are synchronized with the actual content of the various tables using these OIDs. This is required, for example, after (re-)importing FraudOne data from a backup or database copy action. The FraudOne server system must be offline while this task is being executed.
<code>--resetusermanagement</code>	Deletes all user and group records and the associated access rights and group membership information from the user management tables and then sets up the standard FraudOne default master user for the authentication model chosen in the SBDebTab.properties file. Often used to switch the authentication model between FraudOne's native model and the Active Directory integration model.
<code>--<TABLESELECTIONNAME></code>	Selects the named table (one from the lists given above) to be created or modified. If no table names are selected, all tables are processed. More than one table can be selected for any execution of the program.

Property file options

The available file options are listed in the following table:

Key	Description
<code>DisplayOnly =</code>	<p>Yes - The DDL generated by the program is not executed but only listed on the standard output (it can be redirected to a file for easier reading or editing).</p> <p>No - The program will connect to the database and execute the generated DDL immediately.</p> <p>Possible values: Yes No</p>
<code>TargetDBMS =</code>	<p>Possible values: DB2ZOS DB2UDBSBCS DB2UDBUNICODE ORACLE MSSQL</p> <p>DB2ZOS - is used for DB2 installation on a mainframe.</p>

Key	Description
	<p>DB2UDBSBCS - is used for a DB2 installation on a Linux, Unix or Windows system configured to use a non-UTF code page.</p> <p>DB2UDBUNICODE - is used for a DB2 installation on a Linux, Unix or Windows system configured to use a UTF code page.</p> <p>ORACLE - is used for an ORACLE installation.</p> <p>MSSQL - is used for a Microsoft SQL Server installation.</p>
OracleJDBCType =	<p>For TargetDBMS=ORACLE only.</p> <p>TNS - is used to indicate that the database name is a local TNS name (server and port are then not required).</p> <p>OCI - requests an OCI connection to a database at a specified server and port (requires a local Oracle client installation).</p> <p>THIN - uses a "thin client" connection to a database at a specified server and port.</p> <p>Possible values: TNS OCI THIN</p>
DBSchema =	<p>Names the schema to be used for the database tables and supporting structure elements. Only one schema name is supported for each database type (SignBase and SignCheck). (So it is not supported to construct some elements using table selection with one schema, and other elements with another.)</p>
DBUpdateGroup =	<p>The name of a user, group or role which is to be granted the rights needed (by a FraudOne technical user of the database used by the programs accessing the database) to use the database in production. The program grants the rights to SELECT, INSERT, UPDATE and DELETE rows in each table and EXECUTE each stored procedure that is created by the program.</p>
NoStop =	<p>No - The program halts with a prompt before modifying the database.</p> <p>Yes - The program does not prompt before modifying the database. This is intended for unattended operation.</p> <p>Possible values: Yes No</p>
BankName =	<p>Configures the name displayed in the program banner text.</p>
SuperUserId =	<p>The FraudOne user or group identifier to be granted the access rights needed to perform as the initial master user (or group of users). For a FraudOne native authorization model installation this has the default SOFTPRO for backward compatibility. For an Active Directory integrated installation there is no default.</p>

Key	Description
SuperUserIsGroup = Yes No	The identifier provided in the SuperUserId parameter is a user- (No) or a group identifier (Yes). ("Yes" is allowed only with the Active Directory integrated authentication model.)
UseActiveDirectory = Yes No	Controls whether the installation uses the FraudOne native authentication model (No) or the Active Directory integrated authentication model.
GeneratePrimaryIndexes =	For TargetDBMS = DB2ZOS only. Yes - The generated DDL includes statements to create primary indexes for the selected tables. No - For use when DB2 creates primary indexes automatically, no DDL statements are generated to create primary indexes. Possible values: Yes No
SuperUserId =	The FraudOne user or group identifier to be granted the access rights needed to perform as the initial master user (or group of users). For a FraudOne native authorization model installation this has the default SOFTPRO for backward compatibility. For an Active Directory integrated installation there is no default.
SuperUserIsGroup = Yes No	The identifier provided in the SuperUserId parameter is a user- (No) or a group identifier (Yes). ("Yes" is allowed only with the Active Directory integrated authentication model.)
UseActiveDirectory = Yes No	Controls whether the installation uses the FraudOne native authentication model (No) or the Active Directory integrated authentication model.
GenerateAuxiliaryTables =	For TargetDBMS=DB2ZOS only. Yes - The generated DDL includes statements to create the auxiliary tables needed to store LOB data. No - For use when DB2 creates the required auxiliary tables automatically, no DDL statements are generated to create auxiliary tables. Possible values: Yes No
IndexesUsingVCAT = IndexesUsingSTOGROUP =	For TargetDBMS = DB2ZOS only. Only one of the two options should have "Yes" specified. Specifying both as "No" results in no storage assignment clauses for indexes being generated. IndexesUsingVCAT = Yes results in a "using vcat" clause being included in the DDL generated to create an index. IndexesUsingSTOGROUP=Yes results in a "using stogroup" clause being included in the DDL generated to create an index.

Key	Description
	Possible values: Yes No
AllowGenerateUnique =	<p>For TargetDBMS=SB2ZOS only.</p> <p>If the target DB2 system supports the function "timestamp(generate_unique())" then "Yes" should be specified for this option to ensure that the timestamps inserted into various tables by the program are unique in those cases where uniqueness is required.</p> <p>Specifying "No" causes the program to use "current timestamp" in place of the "timestamp(generate_unique())" function. In this case, non-unique timestamps may be provided by DB2 leading to "duplicate key" errors inserting some fixed rows. If such errors occur, rerunning the program may help.</p> <p>Possible values: Yes No</p>
DefaultTablespace = DefaultIndexTablespace = DefaultLongTablespace =	<p>These three options each identify a table space to be used when no table specific table spaces (see below) are defined for a table.</p> <p>DefaultLongTablespace= identifies a table space which will contain BLOB data for all tables where no table specific table space is defined. If no long table space is identified, the BLOB data will be placed into the table space identified by "DefaultTablespace=".</p> <p>DefaultIndexTablespace= identifies a table space which will contain index data for all tables where no table specific table space is defined. If no index table space is identified, the index data will be placed into the table space identified by "DefaultTablespace=".</p> <p>DefaultTablespace= identifies a table space which will contain the table data for all tables where no table specific table space is defined. This table space is also used for index data and BLOB data when no specific table space for these data types is defined.</p>
DefaultIndexBufferPool =	<p>For TargetDBMS=DB2ZOS only.</p> <p>This causes a "bufferpool" clause to be added to the DDL statement creating an index containing the value specified here.</p> <p>The value for DefaultIndexBufferPool is used for all indexes for all tables where no table specific buffer pool is defined.</p>
<TABLENAME>Tablespace = <TABLENAME>IndexTablespace = <TABLENAME>LongTablespace =	<p>These three options each identify a table space to be used when creating the named table. The table names are taken from the lists above.</p> <p>If NO table space is identified specifically for a table, the table spaces defined using the "Default..." options above are used for the table. If also no default table spaces are defined, the DDL is generated without storage referencing clauses.</p>

Key	Description
	<p><TABLENAME>LongTablespace= identifies a table space which will contain BLOB data for the table. If no long table space is identified, the BLOB data will be placed into the table space identified by "<TABLENAME>Tablespace=".</p> <p><TABLENAME>IndexTablespace= identifies a table space which will contain index data for the table. If no index table space is identified, the index data will be placed into the table space identified by "<TABLENAME>Tablespace=".</p> <p><TABLENAME>Tablespace= identifies a table space which will contain the table data for the table. This table space is also used for index data and BLOB data when no specific table space for these data types is defined.</p>
<TABLENAME>IndexBufferPool =	<p>For TargetDBMS=DB2ZOS only.</p> <p>This causes a "bufferpool" clause to be added to the DDL statement creating an index containing the value specified here.</p> <p>This value is used for all indexes for the selected table. If no value is provided here, the value of "DefaultIndexBufferPool" is used. If also no default value is provided, no bufferpool clause is added to the DDL statement creating the index.</p>

DB2 database

Database code page

The DBA should select a code page for the database that can represent all national language characters needed by their use of FraudOne.

In extreme cases, a UTF code page can be used for the database when no single byte code page can be found that satisfies this requirement.

Selecting a UTF code page has an adverse effect on the physical disk space needed for the database however this is a simple functional decision, if UTF is required then the additional disk space is also required.

Once a code page is selected the (SYSTEM and/or DB2) environment entry DB2CODEPAGE in the Control Panel and/or DB2 registry parameters (DB2SET) should be set to match the selection.

In order to create the database with the Command Line Processor, enter the following command:

```
CREATE DATABASE <dbname> ON <drive> USING CODESET <code page>
TERRITORY <nations-flag>
```

After the creation of the databases

The **Database Manager Configuration** parameter `QUERY_HEAP_SIZE` must be at least **2500** for SignCheck and SignInfo. A lower value could be set up only for the SignBase database. The setting can be changed using DB2 configuration tools.

In order to use the DB2 CLP window enter the following command:

```
db2 UPDATE DBM CFG USING QUERY_HEAP_SZ 2500
```

The **Database Configuration** parameter `LOCKLIST` has a default value of **50** which can be too low for SignCheck installations where a high number of transactions are being processed. If the SignCheck Getter, Putter or Workflow applications report deadlocks and the DB2 diagnostic log or DB2 performance monitor shows that these deadlocks are due to lock escalation, we recommend increasing the value of this parameter to **5000**; this requires an additional 20MB of main memory over the required default value. This setting can be changed using DB2 configuration tools.

If you wish to use the DB2 CLP window enter the following command:

```
db2 UPDATE DB CFG FOR <dbname> USING LOCKLIST nnnn
```

Authentication and authorization

For DB2 the groups and users used to authenticate to DB2 are simple operating system users.

Kofax recommends using a group identifier for the assignment of authorization to access the data in the FraudOne tables. Any users (including the technical user(s) for the various FraudOne programs) that require access to the FraudOne data can be simply added to the group where the authorization has already been done.

Placing the name of this group into the properties files of the table creation programs will ensure that this group is granted the required authorizations to access the FraudOne data.

One additional authorization is needed for this group; it must be allowed to "use" all table spaces where the FraudOne data is stored and this must be done by the DBA at the time each table space is created:

```
grant use of tablespace <tablespacename> to group <groupname>;
```

For "trusted client" authentication Kofax supports and recommends only the use of Active Directory (plus some external Kerberos implementations for Linux or Unix database servers). To enable this it is necessary to change the database manager configuration to use "AUTHENTICATION=KERBEROS". All systems involved must be members of the same domain or share a trust relationship with each other.

To enable the trusted connection for a DB2 database management system configuration, it is necessary to use

```
Userid=*
```

and/or

```
UseridSC=*
```

in the **SrvMgr4.ini** configuration file for the application servers.

Maintenance of the database

During the recording phase it is very important that the increasing database will be frequently reorganized. Up-to-Date statistics for all the tables are important to give the DB2 optimizer the proper indications on choosing the access path.

This is normally configured to be done as part of DB2's automatic maintenance but it can also be done manually using the following commands:

```
DB2 CONNECT TO <dbname>

DB2 REORGCHK UPDATE STATISTICS ON TABLE ALL
DB2 CONNECT RESET
```

After the maintenance, it is necessary to restart each FraudOne server program before the newly gathered statistics will be effective in creating database access plans.

ODBC configuration

Once the database is created and populated, it is necessary (on each system where a FraudOne server program will be used) to create ODBC data sources linking to each database to be used from that system. Creating such a data source entry requires that the target database(s) be cataloged by the DB2 client installation.

No special parameters are needed in the ODBC configuration.

DB2 on IBM mainframe systems

The **SBDebTab** and **SCDebTab** procedures support the creation of the required tables on IBM mainframe systems using DB2 for z/OS. The procedure requires access to the mainframe DB2 through DB2 Connect.

As always, the contents of the properties files must be set up correctly in order to generate working DDL for the table creation.

A number of additional parameters are needed to configure the programs correctly for DB2 on a mainframe system. These are described together with the other options in the table above.

Oracle database

For Oracle products, note that only Oracle 10g or higher is supported.

ODBC settings

FraudOne requires that the "Numeric settings" option for the Oracle ODBC driver is set to the value "Oracle NLS settings" (or the equivalent in your installation language).

Database code page

FraudOne supports the use of any database code page.

Number of open cursors

The maximum number of open cursors ("open_cursors" in **init.ora** for the FraudOne Database) per session should be set to be higher than **1000**.

```
Init.ora:
[...]
```



```
open_cursors=1000  
[...]
```

Authentication and authorization

Notice that ALL users accessing an Oracle database need the authorization "CREATE SESSION". This is in addition to the authorizations described below.

Oracle requires that a user be created for the schema name selected for the FraudOne application's tables. This user need not be enabled to connect to the database. It must, however, have a usage quota defined for every table space used to contain FraudOne data.

```
create user "<schema>" identified by "<schema>" profile "DEFAULT"  
  account lock  
  default tablespace "<tablespacename>"  
  temporary tablespace "TEMP";  
alter user "<schema>" quota unlimited on "<tablespacename>";
```

The administrative user used to execute the table creation programs needs enough authorizations to ensure that the tasks carried out by the programs will succeed. The following list shows the authorizations needed (assuming the user to be used is SPADMIN).

```
grant CREATE SESSION to "SPADMIN";  
grant DROP ANY TABLE to "SPADMIN";  
grant REDEFINE ANY TABLE to "SPADMIN";  
grant DROP ANY TRIGGER to "SPADMIN";  
grant ALTER ANY TABLE to "SPADMIN";  
grant CREATE PROCEDURE to "SPADMIN";  
grant LOCK ANY TABLE to "SPADMIN";  
grant CREATE ANY VIEW to "SPADMIN";  
grant ALTER ANY INDEX to "SPADMIN";  
grant IMPORT FULL DATABASE to "SPADMIN";  
grant DROP ANY INDEX to "SPADMIN";  
grant CREATE ANY INDEX to "SPADMIN";  
grant ALTER ANY TRIGGER to "SPADMIN";  
grant DROP ANY SEQUENCE to "SPADMIN";  
grant CREATE ANY SEQUENCE to "SPADMIN";  
grant CREATE ANY TRIGGER to "SPADMIN";  
grant ALTER SESSION to "SPADMIN";  
grant EXPORT FULL DATABASE to "SPADMIN";  
grant ALTER ANY SEQUENCE to "SPADMIN";  
grant DROP ANY VIEW to "SPADMIN";  
grant CREATE TRIGGER to "SPADMIN";  
grant CREATE VIEW to "SPADMIN";  
grant EXECUTE ANY PROCEDURE to "SPADMIN";  
grant DROP ANY PROCEDURE to "SPADMIN";  
grant ALTER ANY PROCEDURE to "SPADMIN";  
grant CREATE TABLE to "SPADMIN";  
grant CREATE SEQUENCE to "SPADMIN";  
grant CREATE ANY TABLE to "SPADMIN";  
grant GRANT ANY OBJECT PRIVILEGE to "SPADMIN";  
-- and: Needed only to insert initial Admin user and rights  
grant "SIGNPLUSDB" to "SPADMIN";
```

Kofax recommends the use of a Role to assign access rights to the FraudOne data held in the database. Single users can be then granted this Role as authorisation to access the FraudOne data.

```
create role "<rolename>" NOT IDENTIFIED;
```

and then

```
grant "<rolename>" to "<userid>";
```

Using the <rolename> in the DBUpdateGroup option of the table creation program properties file:

```
DBUpdateGroup=<rolename>
```

This ensures that the required authorization is granted to the tables that are created by the programs.

Table space

Table spaces can be created using commands like

```
CREATE SMALLFILE TABLESPACE "SIGNBASE"  
  DATAFILE 'V:\app\O12CADMIN\oradata\DEM42KWA\SIGNBASE.DBF'  
  SIZE 10M AUTOEXTEND ON NEXT 50M  
  LOGGING  
  DEFAULT NOCOMPRESS  
  ONLINE  
  EXTENT MANAGEMENT LOCAL AUTOALLOCATE  
  SEGMENT SPACE MANAGEMENT AUTO;
```

The names used in the create commands for the table spaces can be entered into the properties file options directly.

Maintenance of the database

Up-to-Date statistics for all tables are important in order to provide the Oracle optimizer the correct indication in order to choose the best and fastest access path.

In order to get detailed information about the tables and to generate new statistics, you have to set up the following commands in **sqlplus**:

```
begin  
  dbms_stats.gather_schema_stats('<schema>');  
end;  
/
```

The generated statistics are used for the access path for Application servers and Service Programs, when they are restarted.

Oracle ODBC settings

Configure the ODBC fetch buffer to have size **0**. Otherwise, performance may degrade and some FraudOne features like the SignCheck lists in the Java Client will not work.

Microsoft SQL Server database

Database code page

FraudOne supports the use of any database code page.

Disk space for the database

SQL Server assigns disk space using "filegroups" and files assigned to these file groups. Once a file group is created, the name of the file group can be used in the table creation program properties files as if it was a table space.

A file group can be created as part of the database creation command:

```
create database FR142ELA
on primary (name = DPrimary, filename = 'V:
\FR142ELA\DPrimary.mdf', size = 10 MB, maxsize = unlimited,
filegrowth = 10 MB) ,filegroup SignPlus (name = DSignPlus1,
filename = 'V:\FR142ELA\SSignPlus1.ndf', size = 10 MB, maxsize =
unlimited, filegrowth = 10 MB) ,
(name = DSignPlus2, filename = 'W:\FR142ELA\SSignPlus2.ndf', size
= 10 MB, maxsize = unlimited, filegrowth = 10 MB)
log on (name = LSignPlus1, filename = 'V:\FR142ELA\SSignPlus.ldf',
size = 10 MB, maxsize = unlimited, filegrowth = 10 MB)
collate latin1_general_100_CS_AS;
```

or using the "alter database / add filegroup" and "alter database / add file" commands:

```
alter database FR142ELA add filegroup SB42CLCKDTS;
alter database FR142ELA add file (name = FCLCKDTS1, filename = 'V:
\FR142ELA\FCLCKDTS1.ndf', size=1MB, maxsize=unlimited,
filegrowth=5MB) to filegroup SB42CLCKDTS;
alter database FR142ELA add file (name = FCLCKDTS2, filename = 'W:
\FR142ELA\FCLCKDTS2.ndf', size=1MB, maxsize=unlimited,
filegrowth=5MB) to filegroup SB42CLCKDTS;
```

Schema names

The schema name chosen for the FraudOne database objects must be created and the user that will be creating the database objects needs permission to alter this schema:

```
create schema <schemaname>;
go
grant alter on schema :: <schemaname> to <adminuser>;
go
```

Authentication and autorisation

Users accessing an SQL Server database can be either normal Windows users (or groups) or they can be defined as database users within (and authenticated by) the database server alone. Kofax recommends the use of normal Windows users.

Authorization of technical users to access the FraudOne data (for example for the Application Server programs using ODBC) is most easily done using a Windows user group. A database user can be created for the group and the authorizations to access the FraudOne data tables automatically granted to the group user by the table creation program.

If the group is SignPlusDB in the domain ADDEV1, the user can be created by the commands:

```
use [master];
go
create login [ADDEV1\SignPlusDB] from windows;
go
use [<signplus-database>];
```

```
go
create user [SignPlusDB] for login [ADDEV1\SignPlusDB];
go
```

Another possibility is to create a database role and then grant that role to each database user that must be authorized:

```
USE [<signplus-database>]
go
CREATE ROLE [SignPlusDB]
GO
```

For both options, the table creation program configuration item:

```
DBUpdateGroup=SignPlusDB
```

then grants the required authorizations automatically.

Any technical users can then be created as Windows (or Active Directory) users and either

- making these users members of the group or
- creating database users for them and authorizing these users to the defined role

will grant them the Authorization needed to access the FraudOne data tables.

Trusted client authentication is supported and can be enabled using the **SrvMngr4.ini** configuration parameters

```
Userid=*
```

and/or

```
UseridSC=*
```

SQL Server ODBC settings

Using ODCB driver the following settings have to be selected (3rd configuration page) when installing:

- Check "Use ANSI quoted identifiers"
- Uncheck the "Use ANSI nulls, paddings and warnings"

SignBase and SignCheck database validation

The Database Validation program performs a comprehensive consistency check and correction of all the tables in the SignBase and SignCheck databases. It also reports any consistency errors that are found. The program is configured by means of the **Validation.properties** file.

Important It is strongly recommended that the Validation script be used to correct a database only under direction from Kofax.

Before running the Validation script to correct a database, it is REQUIRED that a full database backup be taken and that the database should not be accessed by other programs during the time the Validation script is running.

The Validation program is started from the command line using one of the supplied command files (or an equivalent) and requires the following syntax:

```
Validation database-name [userid password]
```

The program is a Windows 32 bit console mode application and produces a progress indication through the Default output file, as well as a report file.

When the Validation script is used to correct the database, the changes made to the database are listed in the report file which can be found in the current directory where the script is being run.

Important The validation process requires a significant amount of time and database resources.

It is strongly recommended that this validation be done only for specific tables (see the Testxxxxxx properties below) and only when it is suspected that a data inconsistency is present.

Validation.properties description table:

Key	Description
DBType=	To select the type of database to be accessed Possible values: DB2, ORACLE, MSSQL
DBServer=	Server-identification (SYBASE only, no longer supported)
Tablexxxxxx	The Tablexxxxxx (e.g. TableCustomer) properties identify the tables that are present in the installation and that should be used as a data source when verifying other tables. For example, setting TableAccount=0 will prevent validity checking of the account numbers used in rules.
Testxxxxxx	The Testxxxxxx (e.g. TestCustomer) properties select the tables whose validity is to be tested. For example, setting TestSignatory=0 will disable all tests for the SIGNATORY table. Tables which are marked as "not present" using the Tablexxxxxx properties will not be tested even if the Testxxxxxx is set to 1.
MakeCorrections	Controls if the validation script will correct (Yes) or not (No) the consistency errors that it finds.
ASVDefault	Specifies the replacement value used when the Validation script corrects an invalid ASV flag (only used for corrections) Possible values: 0, 1
ImageStatusDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the ACCOUNT_IMAGE STATUS field (only used for corrections) Possible values: 0, 1, 4
ImageSourceDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the ACCOUNT_IMAGE SOURCE field (only used for corrections) Possible values: 0, 1, 2, 3, 4

Key	Description
ImageQualityDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the ACCOUNT_IMAGE QUALITY field (only used for corrections) Possible values: 0, 1
ImageFilteredDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the ACCOUNT_IMAGE FILTERED field (only used for corrections) Possible values: 0, 1, 2
MaskBackgroundDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the MASK BACKGROUND field (only used for corrections) Possible values: 0, 1, 2
MaskUseSivalDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the MASK USE_SIVAL field (only used for corrections) Possible values: 0, 1
MaskUseProperDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the MASK USE_PROPER field (only used for corrections) Possible values: 0, 1
MaskUseOCRDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the MASK USE_OCR field (only used for corrections) Possible values: 0, 1
DocumentRotationDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the DOCUMENT ROTATION field (only used for corrections) Possible values: 0, 1, 2, 3
DocumentTypeDefault	Specifies the replacement value used when the Validation script corrects an invalid value in the DOCUMENT TYPE field (only used for corrections) Possible values: 0, 1, 2, 3, A, B, C
Ext<n>Fields	Number of Extension fields for table EXTENSION<n>
Ext<n>Field<f>Type	Data type of Extension field <f> for table EXTENSION<n> Possible values: Char, Date, CharBin, Short, Int, Decimal, String
Ext<n>Field<f>Default	Default value to use for Extension field <f> for table EXTENSION<n> (only used for corrections) Example

Key	Description
	Ext1Fields=2 Ext1Field1Type=String Ext1Field1Default="Hello world" Ext1Field2Type=Int Ext1Field2Default=42

Data Warehouse

The data structures for the Data Warehouse should be placed in a separate database from both the SignBase and SignCheck databases.

Schema names

The schema names for the Data Warehouse are not fixed; they are defined at the time the data structures are created. Depending on the database engine used, it may be necessary to define the schema names before running the installation script.

Preparing the data structures

The tables, views, etc. for the Data Warehouse are created using scripts. These scripts require that the databases, including the tablespaces, exist and that the appropriate command line tools are available.

The scripts are configured to suit the local installation using parameter settings in the file **DWSPCnfg.cmd**, these can be changed using any text editor program.

The configuration file **DWSPCnfg.cmd** must be present and identical in two directories in a standard FraudOne installation:

1. the SignPlus installation directory and
2. the scripts directory that is a child of the SignPlus installation directory.

Configuration entries for all databases

Key	Description
DWDBTYPE=	Database type. Possible values are: DB2UDB (Windows / Unix / Linux) DB2390 (Z/OS) ORACLE MSSQL
DWDBNAME	Data Warehouse database name
DWDBSCHV	Schema name for Data Warehouse data tables

Key	Description
	Default: vmr
DWDBSCHD	Schema name for Data Warehouse public views Default: dwsp
DWDBSCHA	Schema name for Short Term Archive views Default: dwsp
DWDBSCHS	Schema name for Staging Tables Default: vmr
SBDBNAME	SignBase database name
SBDBSCHB	Schema name for SignBase Tables Default: DEB
SCDBNAME	SignCheck database name
SCDBSCHC	Schema name for SignCheck Tables Default: DEB

Configuration entries for DB2 Windows / Unix / Linux

Key	Description
DWDBDWTS	Tablespace for Data Warehouse data tables in the Data Warehouse Database
DWDBDWIS	Tablespace for Data Warehouse data table indexes in the Data Warehouse Database
DWDBSATS	Tablespace for Short Term Archive tables in the Data Warehouse Database
DWDBSAIS	Tablespace for Short Term Archive table indexes in the Data Warehouse Database
DWDBSALS	Tablespace for Short Term Archive table LOB-Data in the Data Warehouse Database
DWDBSBTS	Tablespace for Staging tables for SignBase Data in the Data Warehouse Database
DWDBSBIS	Tablespace for Staging table indexes for SignBase Data in the Data Warehouse Database
DWDBSCTS	Tablespace for Staging tables for SignCheck Data in the Data Warehouse Database

Key	Description
DWDBSCIS	Tablespace for Staging table indexes for SignCheck Data in the Data Warehouse Database
SBDBSBTS	Tablespace for Table DFPKEY in the SignBase Database
SBDBSBIS	Tablespace for Table DFPKEY Index in the SignBase Database
SCDBSCTS	Tablespace for Table DFPKEY in the SignCheck Database
SCDBSCIS	Tablespace for Table DFPKEY Index in the SignCheck Database
DWD2XDIR	Directory for exported data files
DWD2HIST	Target date specification to limit the archive Default: "91 days"

Configuration entries for DB2 Z/OS

Key	Description
DWDBZPRI	Y/N generate primary key indexes
DWDBZIXV	Y/N generate indexes using VCAT syntax
DWDBZIXS	Y/N generate indexes using STOGROUP syntax
DWDBZTB_CI	Tablespace for Short Term Archive table CI
DWDBZTB_CIB	Tablespace for Short Term Archive table CI BACK_IMAGEM
DWDBZTB_CIF	Tablespace for Short Term Archive table CI FRONT_IMAGEM
DWDBZTB_CN	Tablespace for Short Term Archive table CN
DWDBZTB_CR	Tablespace for Short Term Archive table CR
DWDBZTB_ER1	Tablespace for Data Warehouse Holding table ER1
DWDBZTB_FD	Tablespace for Data Warehouse Holding table FD
DWDBZTB_FR	Tablespace for Data Warehouse Holding table FR
DWDBZTB_HN	Tablespace for Data Warehouse Holding table HN
DWDBZTB_HS	Tablespace for Data Warehouse Holding table HS
DWDBZTB_HY	Tablespace for Data Warehouse Holding table HY
DWDBZTB_QM	Tablespace for Data Warehouse Holding table QM

Key	Description
DWDBZTB_SD	Tablespace for Data Warehouse Holding table SD
DWDBZTB_STCN	Tablespace for Staging table STCN
DWDBZTB_STCR	Tablespace for Staging table STCR
DWDBZTB_STCS	Tablespace for Staging table STCS
DWDBZTB_STCW	Tablespace for Staging table STCW
DWDBZTB_STCX	Tablespace for Staging table STCX
DWDBZTB_STCY	Tablespace for Staging table STCY
DWDBZTB_STW21	Tablespace for Staging table STW21
DWDBZTB_UM	Tablespace for Data Warehouse Holding table UM
DWDBZTB_W1	Tablespace for Data Warehouse Holding table W1
DWDBZTB_W10	Tablespace for Data Warehouse Holding table W10
DWDBZTB_W12	Tablespace for Data Warehouse Holding table W12
DWDBZTB_W13	Tablespace for Data Warehouse Holding table W13
DWDBZTB_W15	Tablespace for Data Warehouse Holding table W15
DWDBZTB_W17	Tablespace for Data Warehouse Holding table W17
DWDBZTB_W18	Tablespace for Data Warehouse Holding table W18
DWDBZTB_W2	Tablespace for Data Warehouse Holding table W2
DWDBZTB_W4	Tablespace for Data Warehouse Holding table W4
DWDBZTB_W4C	Tablespace for Data Warehouse Holding table CITIW4
DWDBZTB_W6	Tablespace for Data Warehouse Holding table W6
DWDBZTB_W8	Tablespace for Data Warehouse Holding table W8
DWDBZTB_WT21	Tablespace for Staging table STW21
DWDBZTB_X1	Tablespace for Data Warehouse Holding table X1
DWDBZTB_X2	Tablespace for Data Warehouse Holding table X2
DWDBZTB_X4	Tablespace for Data Warehouse Holding table X4

Key	Description
DWDBZVS_CI	VCAT or STOGROUP for Short Term Archive table CI
DWDBZVS_CIF	VCAT or STOGROUP for Short Term Archive table CI Auxiliary indexes
DWDBZVS_CN	VCAT or STOGROUP for Short Term Archive table CN
DWDBZVS_CR	VCAT or STOGROUP for Short Term Archive table CR
DWDBZVS_ER1	VCAT or STOGROUP for Data Warehouse Holding table ER1
DWDBZVS_FD	VCAT or STOGROUP for Data Warehouse Holding table FD
DWDBZVS_FR	VCAT or STOGROUP for Data Warehouse Holding table FR
DWDBZVS_HN	VCAT or STOGROUP for Data Warehouse Holding table HN
DWDBZVS_HS	VCAT or STOGROUP for Data Warehouse Holding table HS
DWDBZVS_HY	VCAT or STOGROUP for Data Warehouse Holding table HY
DWDBZVS_QM	VCAT or STOGROUP for Data Warehouse Holding table QM
DWDBZVS_SD	VCAT or STOGROUP for Data Warehouse Holding table SD
DWDBZVS_STCN	VCAT or STOGROUP for Staging table STCN
DWDBZVS_STCR	VCAT or STOGROUP for Staging table STCR
DWDBZVS_STCS	VCAT or STOGROUP for Staging table STCS
DWDBZVS_STCW	VCAT or STOGROUP for Staging table STCW
DWDBZVS_STCX	VCAT or STOGROUP for Staging table STCX
DWDBZVS_STCY	VCAT or STOGROUP for Staging table STCY
DWDBZVS_UM	VCAT or STOGROUP for Data Warehouse Holding table UM
DWDBZVS_W1	VCAT or STOGROUP for Data Warehouse Holding table W1
DWDBZVS_W10	VCAT or STOGROUP for Data Warehouse Holding table W10
DWDBZVS_W12	VCAT or STOGROUP for Data Warehouse Holding table W12
DWDBZVS_W13	VCAT or STOGROUP for Data Warehouse Holding table W13
DWDBZVS_W15	VCAT or STOGROUP for Data Warehouse Holding table W15
DWDBZVS_W17	VCAT or STOGROUP for Data Warehouse Holding table W17
DWDBZVS_W18	VCAT or STOGROUP for Data Warehouse Holding table W18

Key	Description
DWDBZVS_W2	VCAT or STOGROUP for Data Warehouse Holding table W2
DWDBZVS_W4	VCAT or STOGROUP for Data Warehouse Holding table W4
DWDBZVS_W4C	VCAT or STOGROUP for Data Warehouse Holding table CITIW4
DWDBZVS_W6	VCAT or STOGROUP for Data Warehouse Holding table W6
DWDBZVS_W8	VCAT or STOGROUP for Data Warehouse Holding table W8
DWDBZVS_WT21	VCAT or STOGROUP for Staging table STW21
DWDBZVS_X1	VCAT or STOGROUP for Data Warehouse Holding table X1
DWDBZVS_X2	VCAT or STOGROUP for Data Warehouse Holding table X2
DWDBZVS_X4	VCAT or STOGROUP for Data Warehouse Holding table X4
DWDBZVS_X4	VCAT or STOGROUP for Data Warehouse Holding table RT
SBDBZTB_DFPK	Tablespace for SignBase table DFPKEY
SBDBZTB_CITK	Tablespace for SignBase table CITIKEY
SBDBZTB_X2	Tablespace for SignBase table X2
SBDBZVS_DFPK	VCAT or STOGROUP for SignBase table DFPKEY
SBDBZVS_CITK	VCAT or STOGROUP for SignBase table CITIKEY
SBDBZVS_X2	VCAT or STOGROUP for SignBase table X2
SCDBZTB_DFPK	Tablespace for SignCheck table DFPKEY
SCDBZVS_DFPK	VCAT or STOGROUP for SignCheck table DFPKEY
DWSPHISTT	Generate trace (Y) or no trace (N) from DWTruncate program
DWSPHISTC	Commit count for delete actions
DWSPHISTD	Target date for deleting archive records Records with sp_date <= (today-DWD2HISTD days) will be removed. Leave blank ("set DWSPHISTD=") for no historization.
DWSPHISTI	Target date for deleting image archive records

Key	Description
	Records with sp_date <= (today-DWD2HIST1 days) will be removed. Leave blank ("set DWSPHIST1=") for no historization.

Configuration entries for SQL Server

Key	Description
DWDBSRVR	Servename for Data Warehouse Database
DWDBDWTS	Tablespace for Data Warehouse data tables in the Data Warehouse Database
DWDBDWIS	Tablespace for Data Warehouse data table indexes in the Data Warehouse DB
DWDBSATS	Tablespace for Short Term Archive tables in the Data Warehouse DB
DWDBSAIS	Tablespace for Short Term Archive table indexes in the Data Warehouse DB
DWDBSALS	Tablespace for Short Term Archive table LOB-Data in the Data Warehouse DB
DWDBSBTS	Tablespace for Staging tables for SignBase Data in the Data Warehouse DB
DWDBSBIS	Tablespace for Staging table indexes for SignBase Data in the Data Warehouse Database
DWDBSCTS	Tablespace for Staging tables for SignCheck Data in the Data Warehouse DB
DWDBSCIS	Tablespace for Staging table indexes for SignCheck Data in the Data Warehouse Database
SBDBSRVR	Server name for the SignBase Database
SBDBSBTS	Tablespace for Table DFPKEY in the SignBase Database
SBDBSBIS	Tablespace for Table DFPKEY Index in the SignBase Database
SCDBSRVR	Server name for the SignCheck Database
SCDBSCTS	Tablespace for Table DFPKEY in the SignCheck Database
SCDBSCIS	Tablespace for Table DFPKEY Index in the SignCheck Database
DWMSXDIR	Directory for exported data files
DWMSHIST	Target date specification to limit the archive

Key	Description
	Default: "-91"

Configuration entries for Oracle

The export function that moves the data to the Data Warehouse uses one or two Oracle transportable tablespaces. These are used only for the staging tables and must be named in the DWDBSBTS, DWDBSBIS, DWDBSCTS and DWDBSCIS parameters. All of the files containing these tablespaces must be identified in the parameters TTSSBFIL and TTSSCFIL.

Key	Description
DWDBDWTS	Tablespace for Data Warehouse data tables in the Data Warehouse Database
DWDBDWIS	Tablespace for Data Warehouse data table indexes in the Data Warehouse DB
DWDBSATS	Tablespace for Short Term Archive tables in the Data Warehouse Database
DWDBSAIS	Tablespace for Short Term Archive table indexes in the Data Warehouse DB
DWDBSALS	Tablespace for Short Term Archive table LOB Data in the Data Warehouse DB
DWDBSBTS	Tablespace for Staging tables for SignBase Data in the Data Warehouse DB
DWDBSBIS	Tablespace for Staging table indexes for SignBase Data in the Data Warehouse Database
DWDBSCTS	Tablespace for Staging tables for SignCheck Data in the Data Warehouse DB
DWDBSCIS	Tablespace for Staging table indexes for SignCheck Data in the Data Warehouse Database
SBDBSBTS	Tablespace for Table DFPKEY in the SignBase Database
SBDBSBIS	Tablespace for Table DFPKEY Index in the SignBase Database
SCDBSCTS	Tablespace for Table DFPKEY in the SignCheck Database
SCDBSCIS	Tablespace for Table DFPKEY Index in the SignCheck Database
DWORXDIR	Directory for exported data files
TTSSBFIL	Names of the files that make up the transportable tablespace for SignBase data
TTSSCFIL	Names of the files that make up the transportable tablespace for SignCheck data

Key	Description
DWORHIST	Target date specification to limit the archive Default: "interval '91' day"

Starting the installation

The script **DWSPDBIn.cmd** is called from the command prompt to create the Data Warehouse data structures. This script needs the user ids and passwords for the three databases that might be accessed:

```
C:\SignPlus\Scripts> DWSPDBIn sbuser sbpass scuser scpass dwuser  
dwpass
```

FraudOne Server Manager

This chapter describes the FraudOne Server Manager in the standard functionality as well as the configurations that can be performed by an administrator. The availability of certain features may vary depending on the products you have purchased.

Overview FraudOne Server Manager

The FraudOne Server Manager is the single control interface for all components in the FraudOne architecture. The Server Manager is responsible for:

- Controlling the Engines and Application servers connected to the FraudOne system. This includes the following:
 - Starting and Stopping Engines and Application servers
 - Restarting crashed servers
 - Provides an interface for the FraudOne Engines and Application servers which enables to write message into the log files
 - Change the trace levels of the log files
- Providing connectivity to the Server Monitor visual client to view the system status
- Start and stop FraudOne programs at pre-configured dates and times

For the remainder of this chapter, the term *client* refers to any process that issues a request to the Attach Manager (visual or non-visual).

This chapter describes the Server Manager and Attach Manager in the FraudOne system.

The Attach Manager provides the load balancing of requests. It distributes the messages received from clients to the appropriate FraudOne Application servers.

The [Attach Manager](#) section in this chapter further describes this component.

All licensing status information can be viewed with the FraudOne Server Monitor. In order to view the licensing information, use a right-click on the desired client and select **Statistics**. Consult your Kofax business representative for further information regarding licensing.

Server Manager, Attach Manager and FraudOne Servers

The following diagram depicts the Server Manager control interface and the Attach Manager in the FraudOne system:

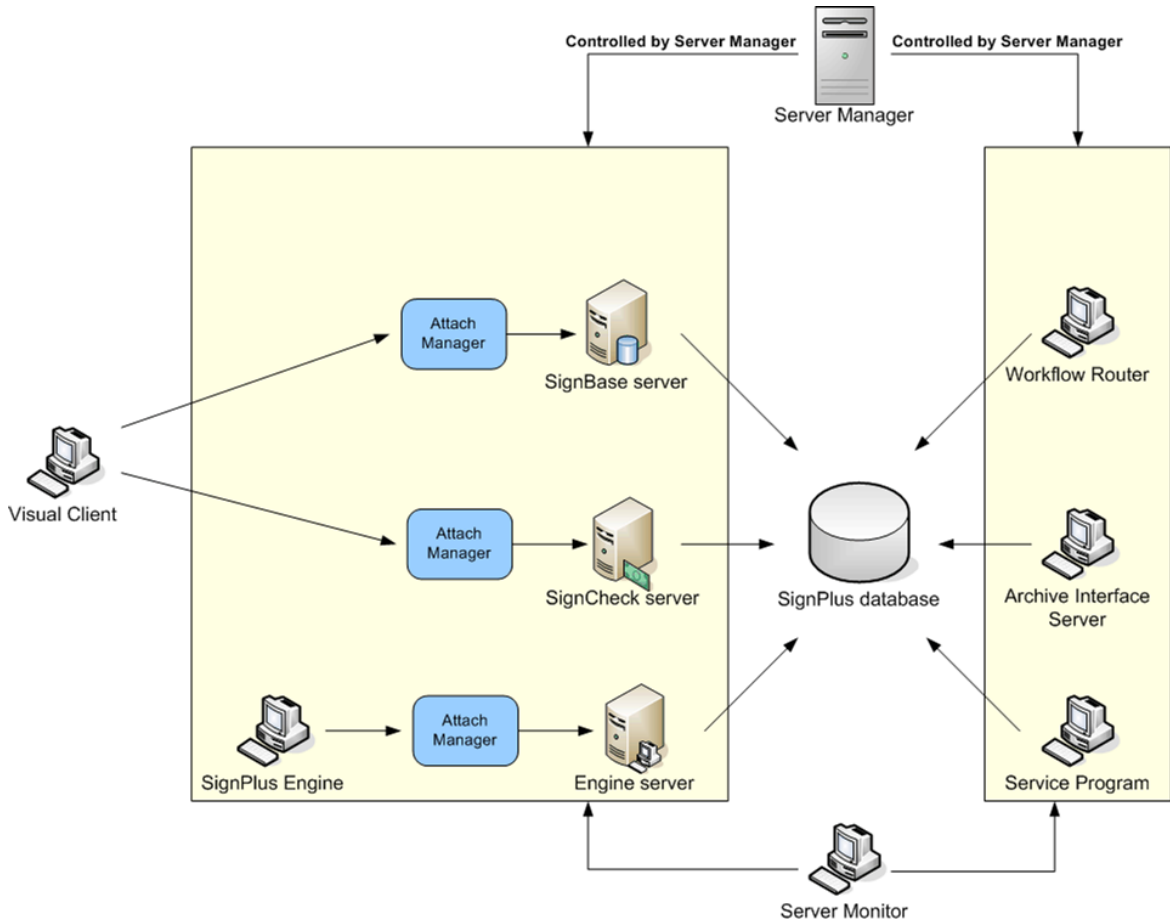


Figure 6: Server Manager control interface in the FraudOne system

Server Manager installation

The FraudOne Server Manager is part of every standard installation. In general, this component will be customized and/or configured by Kofax for the end user's business needs. The availability of certain features may vary depending on the products you have purchased and your system configuration.

See *Kofax FraudOne Installation and Migration Guide* regarding proper Server Manager installation procedures.

Server Manager configuration

The configuration data for the FraudOne programs is held as part of a SignBase database in the form of topography records and configuration file elements.

One topography record exists for each machine (or virtual machine) on which a FraudOne component is installed. The data in this record controls which program types the Server Manager will start on the machine and how many. A topography record can be created only using the **SetupCfg.cmd** script. This creates an "empty" topography record for the machine on (or for) which it is used. The contents of the topography record are managed using the Administration Client.

When a topography record (for a server system) is changed, the corresponding Server Manager must be restarted before the changes are effective.

The configuration information for each FraudOne program instance is stored in at least one configuration file element (one for each configuration "file" needed by the program). The most common file element is the one for SrvMgr4.ini. In the database, in contrast to the original file solution, each file element is stored and edited separately even when the same file name is used.

Each file element consists of a private component that corresponds to the previous configuration section and a shared (common) component that corresponds to the previous common section. Each of these file elements is assigned to a specific instance of a program by means of the address mapping that is associated with the file element.

Using the address map it is possible to assign the same file element to more than one process and, in particular, the same common component to more than one file element. Best practice is to have a single common component that is used by all file elements.

The configuration parameters held in each file element are described below in the tables describing the parameters for each individual program. If a parameter is changed the program instance or instances that use the changed parameter must be restarted before the changes are effective.

SetupCfg.cmd command options for client installation

When setting up a new system using the **SetupCfg.cmd**, you need to know which type of system you wish to create.

The choices are between

- a simple client system, where no Server Manager is installed
- a server system, where a Server Manager is installed, that will be used as a configuration server
- any other server system

For a client system, the SetupCfg.cmd is used to configure the (TCP/IP) address(es) of the configuration server system(s) that are available to the client.

The command sequence to do this is:

```
SetupCfg.cmd -initclient client-machine-name
SetupCfg.cmd -addserver configuration-server-address:server-port
SetupCfg.cmd -addauthserver authentication-server-address:server-port
```

Option	Description
client-machine-name	Is the unique name of the machine known to the FraudOne configuration management tools and is

Option	Description
	normally a descriptive name not related to the TCP/IP address of the machine.
configuration-server-address authentication-server-address	Are the TCP/IP addresses, respectively, of the machine providing the configuration data services and the user authentication services for the installation.
server-port	Is the port address on the server system where the server Attach Manager is listening.

In both these cases, more than one server can be specified by repeating the command with the additional addresses. The servers will then be used in sequence until one responds.

SetupCfg.cmd command options for configuration server installation

For a server system where a Server Manager is used that will deliver the configuration data services for the installation, the **SetupCfg.cmd** is used to configure the ODBC database connectivity to the database containing the configuration data.

The sequence to do this is:

```
SetupCfg.cmd -initlocal server-machine-name :server-port ODBC-DSN
              schema-name
```

Option	Description
server-machine-name	Is the unique name of the machine known to the FraudOne configuration management tools and is normally a descriptive name not related to the TCP/IP address of the machine.
server-port	Is the port address on the server system where the configuration server Attach Manager should listen.
ODBC-DSN	Is the name used in the ODBC configuration to identify the connection to the database that contains the configuration data.
schema-name	The database schema where the configuration tables (SP_ADDRESSMAP and SP_CONFELEM) are stored.

SetupCfg.cmd command options for other server installation

For a server system where a Server Manager is used that does not deliver the configuration data services for the installation, the SetupCfg.cmd is used to configure the (TCP/IP) address(es) of the configuration server system(s) that are available to the server.

The command sequence to do this is:

```
SetupCfg.cmd -initremote server-machine-name  
SetupCfg.cmd -addserver configuration-server-address:server-port  
SetupCfg.cmd -addauthserver authentication-server-address:server-port
```

Option	Description
server-machine-name	Is the unique name of the machine known to the FraudOne configuration management tools and is normally a descriptive name not related to the TCP/IP address of the machine.
configuration-server-address authentication-server-address	Are the TCP/IP addresses, respectively, of the machine providing the configuration data services and the user authentication services for the installation.
server-port	Is the port address on the server system where the server Attach Manager is listening.

In both these cases, more than one server can be specified by repeating the command with the additional addresses. The servers will then be used in sequence until one responds.

SetupCfg.cmd command options to update the database technical user and password

If the database ODBC DSN, port number, technical user and/or password to be used by the local configuration server must be changed, the following command options are used to carry out the configuration update.

```
SetupCfg.cmd -updatelocal [:server-port] [ODBC-DSN]  
SetupCfg.cmd -reauthenticate
```

The command `-reauthenticate` is used when only the user-id and password must be changed without affecting any other system configuration option.

Option	Description
server-port	Is the unique name of the machine known to the FraudOne configuration management tools and is normally a descriptive name not related to the TCP/IP address of the machine.

Option	Description
ODBC-DSN	Are the TCP/IP addresses, respectively, of the machine providing the configuration data services and the user authentication services for the installation.

The technical user and associated password needed can be provided in the following GUI dialog prompt.

SetupCfg.cmd command options for remote installation

Normally, **SetupCfg.cmd** is used locally on the system being set up. When used in this way, the program creates a file called **SPCfInst.dat** in a product-specific directory under the "All Users Profile" directory. Typically this is:

```
C:\ProgramData\kofax\fraudone\SPCfInst.dat
```

In cases where this is impractical, the program can be instructed to perform the setup "as if" it was being run on some other machine and so carry out the setup for that other machine without physically requiring a presence on that machine. In this case, it is the responsibility of the customer administrator to move the created file into the correct location on the target system, typically using a software distribution tool.

The additional SetupCfg.cmd option "for machine-name" is used for this, as follows:

```
SetupCfg.cmd for client-machine-name -initclient client-machine-name  
SetupCfg.cmd for client-machine-name -addserver configuration-server-  
address:server-port  
SetupCfg.cmd for client-machine-name -addauthserver authentication-  
server-address:server-port
```

Notice that the two occurrences of client-machine-name in the -initclient must match. The "for" command option can be used for any of the SetupCfg.cmd "-init" options.

This command sequence results in the correct topography records for the target machine being created and a configuration file called "client-machine-name.dat" being created in the local machine's "All User Profile" product directory. It is this file that must be moved to the target machine and renamed "SPCfInst.dat".

SetupCfg.cmd command options for configuration management

The SetupCfg.cmd program is also used to assist in the initial configuration of the complete FraudOne installation.

There are two command options provided for this purpose, "-migratesrvnmgr" and "-importworkspace".

"-migratesrvnmgr" is provided ONLY as an assistance in loading the configuration information into a FraudOne configuration database from an existing **SrvMngr4.ini** file. The result contains all of the information from the INI file copied one to one into database configuration elements without any optimization or reorganization.

Typically this does not deliver a working configuration!

This option is intended only as an editing helper to load the data. The results must be manually edited to be used in production.

"-importworkspace" is used when a completely new FraudOne system is delivered to load (at least) the basic core configuration of the Administration Client into the configuration database. It is also used with the option "refresh" when the core and sample properties of an existing installation are being upgraded – for example to a new release or as part of a fix pack.

When a FraudOne installation is delivered, it will contain an exported configuration workspace that must be loaded into the configuration database before any further actions can be carried out. This will include the basic core configuration data for the Administration Client and may, at Kofax' option, include further preconfigured software elements.

To load this data into the configuration database, the following actions are needed:

- Use the "-initlocal" command described above to initialize a configuration server system.
- On that system, start the SrvMngr4.exe program. (This results in a working configuration server.)
- On the same system, use the "-importworkspace" to import the provided initial configuration data.
- Without stopping the configuration server ...
- Use the Administration Client to complete the configuration of the FraudOne installation.
- (Re)start the server systems that have been configured.

Using the Server Manager

This section covers the following Server Manager topics:

- Starting the Server Manager using the executable file (**SrvMngr.exe**)
- Starting the Server Manager as a Windows service
- Starting and stopping FraudOne servers using the Server Manager's scheduling interface
- Using the FraudOne Scheduler feature (Windows only)

Starting Server Manager

The Server Manager can be started by using the executable file located in the FraudOne installation directory.



To start the Server Manager double-click the **SRVMNGR4** icon, located in the FraudOne installation directory.

A windows command window will open in which the program will be running. Log output will be displayed in the command window. During its execution lifetime, the Server Manager will continue to output log information to this window.

```

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\FraudOne\SignPlus>srvmngr4
ClientDong: DemoBank
set PATH %shared%;c:\FraudProtect;c:\j2sdk1.4.2_05\bin;c:\PROGRAM FILES\THINKPAD\UTILITIES;c:\WINDOWS\system32;c:\WINDOWS\C:\WINDOWS\System32\Win
ol Panel;c:\Program Files\ATI Technologies\Fire GL 3D Studio Max;c:\Program Files\ATI Technologies\Fire GL Control Panel;S:\DEU\DEUTOOLS\CUS\NT;
\BIN;c:\IBM\SQLLIB\BIN;c:\IBM\SQLLIB\FUNCTION;c:\IBM\SQLLIB\SAMPLES\REPL;c:\Program Files\Support Tools;c:\wslick\win;c:\Installations\FNB\Rel
s\WebSphere\apps\server\installedapps\paneuropan\SignPlus_war\ear\SignPlus_war\WEB-INF\classes;
License files to be used
Manufacturer : SOFTPRO Software Professional GmbH & Co. KG
Customer : Demo
Inst Code : 9430-BF2D-3670-5BC5-FD76
Product : SignBase
SRVMNGR4: SRVMNGR4.INI, (c) SOFTPRO
MonRequestPort = 2004
MonLogPort = 2005
D 2005-07-07 10:26:11.4 SrvMngr4 -- DAULIB.DLL: File:c:\FraudOne\SignPlus\shared\DAULIB.DLL, 06-06-2005 17:43:18, 433.628 bytes, Ver: 1.00.30.0.
D 2005-07-07 10:26:11.4 SrvMngr4 -- TCPACK.DLL: File:c:\FraudOne\SignPlus\shared\TCPACK.DLL, 28-04-2005 14:47:30, 119.260 bytes, Ver: 1.00.10.0.
D 2005-07-07 10:26:11.4 SrvMngr4 -- SPSURS.DLL: File:c:\FraudOne\SignPlus\shared\SPSURS.DLL, 28-04-2005 14:46:08, 199.132 bytes, Ver: 1.01.01.0.
Go KG
D 2005-07-07 10:26:11.5 SrvMngr4 -- FSXTKRM.DLL: File:c:\FraudOne\SignPlus\shared\FSXTKRM.DLL, 19-05-2005 15:45:46, 346.588 bytes, Ver: 3010.
D 2005-07-07 10:26:11.7 SrvMngr4 -- FSXTK.DLL: File:c:\FraudOne\SignPlus\shared\FSXTK.DLL, 19-05-2005 15:46:10, 238.556 bytes, Kernel:3010, API:
emoBank
D 2005-07-07 10:26:12.5 SrvMngr4 -- SPDSU.DLL.DLL: File:c:\FraudOne\SignPlus\SPDSU.DLL.DLL, 06-04-2005 12:41:24, 345.052 bytes, Ver: -
D 2005-07-07 10:26:12.5 SrvMngr4 -- ngntapi.dll: File:c:\WINDOWS\system32\ngntapi.dll, 04-08-2004 00:56:44, 14.848 bytes, Ver: -
D 2005-07-07 10:26:12.6 SrvMngr4 -- SpSrvMonSNMP.dll: File:c:\FraudOne\SignPlus\SpSrvMonSNMP.dll, 28-04-2005 14:33:20, 80.348 bytes, Ver: -
D 2005-07-07 10:26:13.6 SrvMngr4 -- SPSCHED.DLL: module not available
D 2005-07-07 10:26:13.6 SrvMngr4 -- starting SrvMngr4, File:c:\FraudOne\SignPlus\SRVMNGR4.exe, 10-06-2005 14:26:06, 395.193 bytes, Ver: 1.36.49.
D 2005-07-07 10:26:13.6 SrvMngr4 -- for DemoBank
D 2005-07-07 10:26:19.0 SC-Server-1 -- starting, Version/Build Id: RSTHServer\CoreHSU-37-21, 04-07-2005 14:32:44
D 2005-07-07 10:26:19.4 SB-Server-1 -- starting, Version/Build Id: RSTHServer\CoreHSU-37-21, 04-07-2005 14:32:44
D 2005-07-07 10:26:19.8 WFRouter -- starting, version: 2.28.1.0, 04-07-2005 14:32:54
D 2005-07-07 10:26:19.8 WFRouter -- WFRouter.exe - file time: 2005-05-02 12:53, file size: 188997
D 2005-07-07 10:26:19.8 WFRouter -- using router DLL version: 2.34.1.1 for customer DemoBank
D 2005-07-07 10:26:18.2 SignBase -- starting, Version: 3.07.10.0.00, 04-07-2005 14:32:41, PID = 688
D 2005-07-07 10:26:18.2 SignBase -- starting, File:c:\FraudOne\SignPlus\attman4.exe, 21-06-2005 15:15:22, 336.744 bytes, Ver: 3.07.10.0.00, 04-
D 2005-07-07 10:26:20.8 ASU-Server-1 -- starting, Version: RSTHServer\CoreHAUT-37-06 (3.05.00.0.03), 04-07-2005 14:32:47
D 2005-07-07 10:26:19.4 SignCheck -- starting, Version: 3.07.10.0.00, 04-07-2005 14:32:41, PID = 4936
D 2005-07-07 10:26:19.4 SignCheck -- starting, File:c:\FraudOne\SignPlus\attman4.exe, 21-06-2005 15:15:22, 336.744 bytes, Ver: 3.07.10.0.00, 04-
D 2005-07-07 10:26:21.1 Autonat -- starting, Version: 3.07.10.0.00, 04-07-2005 14:32:41, PID = 5480
D 2005-07-07 10:26:21.1 Autonat -- starting, File:c:\FraudOne\SignPlus\attman4.exe, 21-06-2005 15:15:22, 336.744 bytes, Ver: 3.07.10.0.00, 04-0
D 2005-07-07 10:26:22.4 APIA -- starting, Version: 3.07.10.0.00, 04-07-2005 14:32:41, PID = 4876
D 2005-07-07 10:26:22.4 APIA -- starting, File:c:\FraudOne\SignPlus\attman4.exe, 21-06-2005 15:15:22, 336.744 bytes, Ver: 3.07.10.0.00, 04-07-2

```

Figure 7: Server Manager Log output window

In order to end the Server Manager, simply enter the **SHUTDOWN** command (in capital letters) in the command window or click the **Ctrl** and **C** buttons simultaneously.

Starting Server Manager as a Windows Service

The Server Manager can also be started automatically as a background task using the Windows operating system's service facility. In order to run the Server Manager as a Windows service you need to make sure that the **SPService.exe** file is located in the same directory location as the following files:

- **Attman5.exe** (Attach Manager executable file)
- **SrvMngr4.exe** (Server Manager executable file)

By default, these files are located in the root directory of your FraudOne installation.

Once you have ensured that the **SPService.exe** file is available, the next step is to configure the Server Manager to be executed as a Windows service. To register the new service, open a command window and change directories to the directory into which you copied the **SPService.exe** program. Next, run the following command from the command prompt (this step requires a user with administrative rights):

```
SPService -install
```

The service can then be found in the Windows Control Panel - Chapter Services as:

```
SignPlus Service
```

From the Windows Control Panel, you can specify whether or not the SignPlus Service should be started automatically or manually after the Windows operating system has booted.

It is usually necessary to define a particular Windows user that is used for running the service based on security settings. This user may also be configured via the Control Panel.

The Windows service can be removed from the Registry by entering the following command in the command window:

```
SPService -remove
```

Command line options for the Server Manager

Regardless of how the Server Manager is started, the following options can be supplied with the command. Notice that these are primarily for emergency use only, a normal FraudOne configuration should start the Server Manager with no additional options on the command line.

- `-instancename <Monitor1>` - the address mapping instance name of this Server Manager
- `-sharedpath <path>` - the path to the directory containing FraudOne's common libraries
- `-xclogconfig <file>` - the name of the XML file used to configure the XcLog component for this server
- `-inifile <file>` - the name of the INI file containing the configuration for all components of this server
- `-bootstrap [Y|N]` - "y" starts this one server in error recovery mode (and so ignores all configuration information in the central configuration database).

Scheduling the FraudOne Application Servers

FraudOne provides a command line interface to start and stop specific FraudOne application servers. This feature can be used to configure certain servers to execute at predefined times, or to have servers shut down during routine maintenance.

The scheduler works in conjunction with the Server Manager. The command line interface is called **SPSchedule.exe** and can be found in the FraudOne installation directory.

In order to activate the Scheduler to work with the Server Manager the following entry must be added to the configuration file element for the ServerManager instance type:

```
Schedule= <the path of SPSchedule.exe>
```

Starting and stopping manually

In order to start or stop an application server process manually, the following has to be entered in the Windows command prompt:

```
SPSchedule START <server-name>
```

```
SPSchedule STOP <server-name>
```

Where the `<server-name>` is the name of the server as specified in the topography record.

Starting and stopping automatically

The Windows AT Internal Scheduler can also be used in order to schedule a "start server" at a predefined date and time.

For example, if we would like to start the SignBase server SB-Server-10 at 8:00am and stop the server at 6:00 pm, then enter the following command:

```
AT 08:00 "SPSchedule START SB-Server-10"
```

```
AT 18:00 "SPSchedule STOP SB-Server-10"
```

For detailed functions of Windows AT Internal Scheduler commands enter

AT-?

in the command prompt. The use of the Server Manager by remote computer using the AT command is also described.

FraudOne Scheduler (Windows only)

The goal of this calendar driven system is to schedule starts and stops of FraudOne programs throughout the days of the week and month. It allows for an optional method which provides the following advantages:

- Can be started by remote with access rights
- Allows a visual display of the of the schedule via the XML file (described below)
- The schedule can be changed dynamically by specifying the different parameters

The calendar system can be defined with the combination of the following parameters:

- Start and Stop time
- Days scheduled
- The type of servers or clients used

The FraudOne Scheduler is configured using the configuration file element instance type "SchedulerServer". The following parameters must be provided for both SchedulerServer and ServerManager instance types and the best practice is that these parameters are present in a common file element component that is shared by both:

```
SchedCache =  
SchedDtdFile =  
SchedPort =  
SchedServer =
```

FraudOne Scheduler File Format Specifications

XML is used as the FraudOne Scheduler file format. This provides an application independent way of sharing the information with the connected Server Managers. The XML calendar file a very detailed processing schedule for all programs controlled by the Server Manager. The Server Manager requests for an updated XML file at predefined times from the Scheduler Server. This is configured in the **<SPSchedule>** which is described in the [Supported Entries](#) section.

The following depicts an [XML sample](#) that is based on the DTD (Document Type Definition) schema described below.

The description of the XML and DTD file format is listed in the following section.

XML Sample Based on the DTD File Format

```
<?xml version="1.0"?>
<!DOCTYPE SPSchedule SYSTEM "SPSchedule.dtd">
<SPSchedule Timestamp="2004092801" Refresh="3600" ErrRefresh="120">

  <SPDay Date="*">
    <Inst Name="Getter">
      <Host>test.pc.softpro.de</Host>
      <Uptime>12:00-04:00</Uptime>
      <Param>Test 1 2 3</Param>
    </Inst>

    <Inst Name="Putter">
      <Host>*</Host>
      <Uptime>04:00-05:00</Uptime>
    </Inst>

    <Inst Name="WFRouter">
      <Host>192.168.1.10</Host>
      <Uptime>03:00-11:00</Uptime>
    </Inst>
  </SPDay>

  <SPDay Date="2004-09-29">
    <Inst Name="Getter">
      <Host>test.pc.softpro.de</Host>
      <Uptime>11:00-19:00</Uptime>
    </Inst>
    <Inst Name="Putter">
      <Host>test2.pc.softpro.de</Host>
      <Uptime>12:00-19:00</Uptime>
      <Param>Test Putter 1 2 3</Param>
    </Inst>
    <Inst Name="WFRouter">
      <Host>test.pc.softpro.de</Host>
      <Uptime>12:00-19:00</Uptime>
    </Inst>
  </SPDay>
</SPSchedule>
```

Supported entries

This section describes the tags that may be specified in the FraudOne Scheduler's configuration file.

<SPSchedule>

This tag determines the settings of the Server Manager's requesting times of the XML file.

Timestamp: The timestamp that identifies the schedule. If the timestamp is higher than the one currently used then the updated XML file is distributed to the Server Manager.

Refresh: The time (in seconds) clients should wait before requesting if a new XML document is available.

ErrRefresh: The time (in seconds) that a client waits between failed attempts to contact the server.

<SPDay>

This tag describes the processing date indicated in the Date parameter. The following are valid entries:

1. "YYYY-MM-DD" ISO format for a specific day
2. "*" for the every day configuration (defaults)
3. "0" for Sunday
4. "1" for Monday
5. "2" for Tuesday
6. "3" for Wednesday
7. "4" for Thursday
8. "5" for Friday
9. "6" for Saturday

<Inst>

This tag describes the FraudOne program instances to be scheduled, as indicated by the Name= parameter. The name parameter must be the same as the one configured in the Server Manager configuration file **SrvMgr4.ini**.

For example, the SignCheck Workflow Router might be displayed as [WFRouter] in the **SrvMgr4.ini**, so the tag should be written as <Inst Name="WFRouter">.

<Host>

This tag identifies the hostname or IP address of the host machine instance. This is based according to the DNS configuration or the IP address respectively. If a host name or address is specified, the entry will only apply to that particular host.

"*" can be used to specify that the entry applies to any host.

<UpTime>

This tag defines the time that the FraudOne instance should run. The uptime is defined in the format "HH:MM-HH:MM". Entering the time "00:00-00:00" indicates that this instance should not run at all, while "00:00-24:00" means this instance will run all day.

<Param>

This tag defines the additional run time parameters for the FraudOne programs. These configurations override the parameters in the **SrvMgr4.ini** file.

The FraudOne Service programs use the following key formats:

Parameter	Feature	Key format
GSI	Getter Stop to park Incomplete files	"GSI-HH:MM"
GPI	Getter Process remaining Incomplete files	"GPI-HH:MM"
PRE	Putter Regular End-Of-Day	"PRE-HH:MM"
PFE	Putter Forced End-Of-Day	"PFE-HH:MM"

DTD file format specifications

```
<!--
*
* SignPlus Scheduling DTD
*
* SOFTPRO GmbH
*
-->

<!--
One schedule contains a list of days
Attributes are:
    Timestamp - the timestamp for this schedule. Use YYYYMMDDNN, where NN is a number
within the day
    Refresh - The time in seconds clients should wait before asking if a new document
is available
    ErrRefresh - The time in seconds clients should wait between failed attempts to
contact the server
-->
<!ELEMENT SPSchedule (SPDay*)>
<!--ATTLIST SPSchedule
    Timestamp CDATA #REQUIRED
    Refresh CDATA #REQUIRED
    ErrRefresh CDATA #REQUIRED
-->

<!--
Schedule for one day. Contains instances of applications to be run.
Attributes are:
    Date - the date of the day. Can be:
        YYYY-MM-DD - a specific day
        * - everyday
        0 - Sunday
        1 - Monday
        2 - Tuesday
        3 - Wednesday
        4 - Thursday
        5 - Friday
        6 - Saturday
-->
<!ELEMENT SPDay (Inst*)>
<!--ATTLIST SPDay
    Date CDATA #REQUIRED
-->

<!--
The instance to be scheduled.
Attributes:
    Host - The fully qualified hostname or ip address of the host the instance runs on
(* for all).
    Uptime - The time period in the format "hh:mm - hh:mm".
    Param - Optional parameter list to pass as command line arguments.
-->
<!ELEMENT Inst (Host, Uptime, Param?)>
<!--ATTLIST Inst
    Name CDATA #REQUIRED-->

<!--
The host on which this instance runs.
Can be '*' for all hosts
-->
<!ELEMENT Host (#PCDATA)>

<!--
The time for which this instance should run.
Same rules as for srvmngr4 apply.
Time format is 'hh:mm-hh:mm'.
-->
<!ELEMENT Uptime (#PCDATA)>

<!--
```

The optional command line parameters to be passed to the application.
-->
<!ELEMENT Param (#PCDATA)>

Administration with the Server Monitor

The FraudOne Server Monitor provides a visual interface and control of the Server Manager functionalities. It enables the viewing and administration of the server-side components. For proper functioning the Server Manager and the Server Monitor must be configured to use the same communication port. Refer to the [Server Monitor](#) chapter for instructions on using the Server Monitor and setting the proper port configurations.

Attach Manager

The Attach Manager is an integral component of the Application servers by providing the load balancing of requests. These requests may be from a visual client such as the Java client, or from another process within the system (non-visual).

The Attach Manager is started by the Server Manager and makes the connection between the clients (e.g. SignBase, SignCheck, ASV, etc.) and the Application servers. The Attach Manager receives messages from clients and distributes them to the appropriate FraudOne Application servers. Refer to the Server Manager communication protocol section below.

When a client initiates a request to one of the FraudOne Application servers it is always routed via the Attach Manager. The Attach Manager places the request in a queue until it can be processed by the next available server. Depending on the configuration, the request may be queued for a pre-configured amount of time. If no available server can process the request after the time has elapsed, the client will receive a timeout message and the timeout is written to the **monitor.log**. See [Timeout](#) section for more information.

Attach Manager installation

The Attach Manager is part of every standard installation. In general, this component will be customized and/or configured by Kofax for the end user's business needs.

The availability of certain features may vary depending on the products you have purchased and your system configuration. See *Kofax FraudOne Installation and Migration Guide* regarding proper Attach Manager installation procedures.

Attach Manager configuration

Configuration entries for the Attach Manager instances are provided in configuration file elements with the instance type "AttachManager". It is required that each Attach Manager have a unique configuration file element since there are a number of configuration parameters that must be unique.

Unlike other server and service programs, the Attach Manager(s) do not have their own entry in the topography record. The number of Attach Manager processes is determined by the use of the server program types that require an Attach Manager. Attach Manager processes are always started when they are needed.

The following settings can be configured for the Attach Manager:

- Start and Stop times of the Attach Manager
- Port connection
- User access settings
- Group Name (e.g. Automat, SignBase, SignCheck)
- Server Timeout

Attach Manager communication protocol

The Attach Manager oversees the communication between the clients and various FraudOne servers. The Attach Manager communicates via the FraudOne Clients (Visual and non-Visual) by way of TCP/IP communication protocol. The communication to the server process is done via a Named Pipe.

These parameters are defined in the **SrvMngr4.ini** configuration file and allows for the following Timeout parameters:

- **PipeTimeout:** The maximum time (in seconds) that the Attach Manager waits after sending a request to an application server. If the time expires and no answer was received from the server the Attach Manager records this incident in the log file and issues "server busy" to the client (as long as the client still waits for the answer).
- **WriteTimeout:** The maximum time (in seconds) from the time that the Attach Manager sends the server's return data until it is received by the client (close socket). If the client does not close the socket in time an error entry is put into the log. This is also referred to as the response timeout.
- **ServerTimeout:** The maximum time (in seconds) that a client request waits for an available server. If after that time all defined servers are busy, the Attach Manager returns a "server busy" message to the client.
- **TCPTimeout:** The maximum time (in seconds) allowed for a client to complete a message transmission to the server (first byte to last byte).
- **ServerPipeRestartTime:** The maximum time (in minutes) that the Attach Manager waits on a new client request to server. This occurs after a Pipe Timeout.

Apart from the parameters described above, the following situations may result in the termination of a server process:

- If no reply is received within the PipeTimeout time from a server that is started by the Server Manager
- If a server does not respond within the Pipe Timeout time established by the 'heartbeat check', which is carried out every five minutes on all active servers.
- Following a system failure each affected server is automatically restarted to a maximum 'crash count' of ten. This 'crash count' is reset to zero after 15 minutes of normal operation.

Timeouts for the different clients

The timeout considerations are put into place in order to facilitate the sending and receiving of messages from the clients to the servers and back. The timeouts are used in order to place a limit on the time that a server waits for a request from a client. If a client opens a connection to a server and does not send the request by the appropriate time (TimeOut) then the Attach Manager performs a timeout.

The setting of the timeout parameters controls the behavior of the following:

1. Time that the Attach Manager waits for an answer from the server.
2. How long the Attach Manager waits for a server to handle a request.
3. How long the Attach Manager waits for an request from the client once the communication has been opened.

Timeouts for visual clients (Java Client, Thin Client, Administration Client)

Clients operated manually are defined as the FraudOne Java and Thin Clients. The timeout settings for these clients are defined in the client installation. The configuration files described in the FraudOne Clients chapter describes the use and configuration of these timeout settings.

As a rule of thumb, the time-out values that have been configured for the clients should always be set higher than the total of the timeouts that have been set for the FraudOne servers in the Server Manager's configuration.

Example

The longest possible client request takes 90 seconds (PipeTimeout)

The maximum wait time until a server is available is 5 seconds (ServerTimeout)

The transfer time over the network is also 5 seconds (WriteTimeout)

Server timeout parameters corresponding to these requirements could be:

```
PipeTimeout = 95
ServerTimeout = 10
WriteTimeout = 10
```

In this example the client timeout should be set to value which is 10 seconds more than the sum of the server timeout settings (95 + 10 + 10). So the recommended client timeout would be 125 seconds (115 + 10).

Timeouts for automatic clients (Engines: ASV, Check Stock, etc.)

Clients that are operated without human intervention such as FraudOne Engines are called automatic clients. When the clients are operated automatically then care should be taken to ensure that the server timeout is lower than the client timeout.

Automatic clients have a default timeout value except for Automatic Signature Verification. The default timeout values do not necessarily fit all situations. For the servers that are servicing the automatic clients, the **PipeTimeout** and **ServerTimeout** should be set to equal or slightly below the timeout value of the client. This is configured in the Server Manager configuration file **SrvMgr4.ini**. This is particularly necessary for systems with a high load. Systems with a low or medium load are not affected.

This avoids a situation where a client would timeout on a request that is still being processed by a server.

Important The reason for a Server Timeout by the Attach Manager may be either by a low number of servers servicing the clients or by the database being used by a program.

Message type lists

Clients use messages to request various different functions of the servers. These different function requests use a "message type" encoded into the message data to identify the function requested. The Attach Manager receives these messages and routes them to a server that can carry out the requested function. In general, all servers linked to an Attach Manager can handle all message types that can be received by that Attach Manager.

It is possible (and sometimes required) to specify specific application server instances that should handle specific message types. This may be done for load balancing reasons (e.g. Restricting long running requests to a subset of servers leaving the rest free for shorter requests) or because of configuration detail differences between the individual application servers (e.g. Routing authentication requests to a server connected to a different database).

The parameter `MessageTypeList` is used to select, for each application server, the list of message types (so, functions) that the specific server can process. Servers with no list can accept all messages that the Attach Manager receives.

The Appendix section includes the different Message Types which are categorized by components.

Centralized configuration for SignCheck components

Configuration information for the SignCheck Engine and Workflow components is stored in database tables where they can be managed centrally, subject to access controls and auditing of changes. The configuration information is stored as part of the SignBase database and access to the information is through the SignBase Application Server.

SignBase Application Server configuration

The feature to store configuration information in the SignBase database is enabled by adding the required message type AA to the message type list for that server.

If SignBase is to manage user authorization for access to the configuration information in the database, the database must also contain the SignBase user management data and the SignBase Application Server(s) handling message AA must have also the

`SignSec = Yes`

parameter in its configuration file element.

Support for multiple SignBase Databases

FraudOne allows a customer installation to use more than one SignBase database. When more than one SignBase database is available, the database is dynamically selected by that application server based on the BNO of the request being processed.

Refer to the 'Using the configuration parameters' section below which describes configuration possibilities and contains the key settings tables.

If SignBase User and Rights management is required for the installation, it must be centrally combined for all BNOs and the access to the one database containing the user definition data must be by means of a separately configured SignBase authorization server.

Machine names

Each machine on which a Server Manager controlled installation is run must be assigned a unique name. This name is restricted to forty characters.

Best practice is to use a self describing name like "ABank-Automat-Servers-1".

Configuration Server addresses

Each machine on which a Server Manager controlled installation is run must "know" the address or addresses of the SignBase Server(s) that can deliver the required configuration data and/or authentication services for that machine.

The address consists of a TCP/IP host name for the machine where the server is installed and the port number to which the server is connected, separated by a colon in the usual way, e.g. "IServ200AA.pc.softpro.de:2000".

More than one SignBase Server address can be supplied to ensure system reliability in the event of the failure of a SignBase Server. If more than one SignBase Server address is supplied, each address is used in turn until a connection to one is successful.

The Automat Server, Workflow Router and CRS processes will wait during startup until one of the SignBase Servers can be contacted to obtain the required configuration data.

Configuration setup program

The machine name and SignBase Server addresses must be configured using the supplied program **SetupCfg.cmd** on each machine where any FraudOne component is run.

This program stores the information into a file stored in the directory

```
%allusersprofile%\kofax\fraudone\
```

with the file name "SPCfglnst.dat". This file should not be modified other than using the **SetupCfg** program.

If the authentication services are supplied by the same server(s) as the configuration data then only the "-addserver" command is needed otherwise the address(es) of the server providing authentication services should be added using the "-addauthserver" command.

See [SetupCfg.cmd command options for client installation](#) and following sections for detailed information about use of the SetupCfg.cmd program.

Server Manager configuration file elements for SrvMngr4.ini

The configuration file elements for file name SrvMngr4.ini, located in the SignBase configuration database enable the configuration of the following components:

- Server Manager
- Attach Manager
- SignBase and SignCheck servers
- Automat servers and clients
- Non-Visual clients (FraudOne Engines ASV, GIA, etc.)

- Service Programs
- The interface used between the Server Monitor and Server Manager
- Workflow Router settings

Each file element stored in the database is used by one or more active program instances. The file element used by a specific program instance is selected using the program instance name (SB-Server-1), instance type (SignBaseServer), machine name (IServ102CB, for example) and the file name (SrvMngr4.ini, of course). The selection is done using the Address Map that can be edited in the Administration Client, as can the actual contents of the file element.

Each file element contains the program specific configuration entries for the program instance and is also linked to a corresponding "common" component (usually used by many file elements) containing those configuration entries that are used globally by all programs. Configuration entries which appear in both program specific and common components are resolved by using the entry in the program specific component.

Important After making changes to any configuration file element or the address map, a restart must take place in order for the changes to take effect.

Settings in the Server Manager configuration file

The following tables list the settings specific to each program that can be managed using the Server Manager and which takes some or all of its settings from the SrvMngr4.ini configuration file elements.

The detailed descriptions of the individual configuration parameters can be found in the alphabetical [FraudOne Parameter List](#) in the appendix.

Server Monitor and Manager [Monitor]

The following configuration parameters can be set for the Server Monitor and Manager in the **[Monitor]** section:

```
Name = Monitor1
LicenceUSBDriveLetter = X
LicenceServerAddress =
LicenceServerPort = 2020
LicenceServerTimeout = 2000
LicenceServerRetries = 2
LogFile = Monitor.log
StatFile = Stat.log
SharedPath =
MonRequestPort = 2004
MonLogPort = 2005
HeartBeat = n
ServerStartDelay = ss
SchedCache = SchedCache.xml
SchedDtdFile = SPSchedule.dtd
SchedPort = 2022
SchedServer = (ip address | name)
ServerManagerInternetNetworkPort = 1999
```

SignBase Application Server

The following configuration parameters can be set for the SignBase Application Server. This uses instance type SignBaseServer.

```
Program = sb1.exe
Group = SignBase
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
ServerTimeout = 40
ShowMode = 0
Userid =
Password =
Database =
SBSchema = DEB
DelayOpenDB = No
SignBase = No
SignSec = Yes
PasswordChangeInterval =
PasswordRetryLimit =
PasswordDuplicateInterval =
PasswordDuplicateCount =
UseProcessCredentials =
UseTechnicalUserCredentials =
ActiveDirectoryTechnicalUser =
ActiveDirectoryTechnicalPassword =
ActiveDirectoryPath =
LoopDelayCycleTime = 3600000
ServiceThreadStack = 200
ReplyPipeSleep = 10
ReplyPipeWait = 30000
SignBase =
Statistics =
StatisticsHold =
LoadHookDLL = N
SBProtocol =
SQLTimeOut =
SQLInterruptRetries =
HistoriseAImages = [Y|n]
PropagateSignatory = [Y|n]
PropagateSignature = [Y|n]
PropagateExtension = [Y|n]
SkipInputSignatory = [N|y]
ForwardOnlyCursors = Y|n
SuppressFlags = 0
SBDynRef = No
VirtualBufferLimit =
MsgTypeList = 10,11,...
InsertHistorised = N
EIMigMode = N
DB2390 = N
ServerId =
RawReadSignatureM = N|y
CSProtocol = Y|n
```

SignBase Attach Manager

The following configuration parameters can be set for the SignBase Attach Manager using the instance type AttachManager and instance name SignBase:

```
Program = attman5.exe
Group = SignBase
Port = 2000
PipeTimeout = 60
WriteTimeout = 30
ServerPipeRestartTime = 30
TCPTimeout = 10
ServerTimeout = 60
ShowMode = 0
JCVersion = Version_Number
TCVersion =
CVBLVersion =
```

SignCheck Application Server

The following configuration parameters can be set for the SignCheck Application Server using instance type SignCheckServer:

```
Program = scl.exe
Group = SignCheck
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
ServerTimeout = 40
ShowMode = 0
Userid =
Password =
Database =
DatabaseSC =
DelayOpenDB = Yes
SignCheck =
VirtualBufferLimit =
VirtualBufferStart =
MsgTypeList = 50,52,...
LoopDelayCycleTime = 3600000
ServiceThreadStack = 200
ReplyPipeSleep = 10
ReplyPipeWait = 30000
InsertHistorised = N
WFCheckRights = N
```

SignCheck Attach Manager

The following configuration parameters can be set for the SignCheck Attach Manager using the instance type AttachManager and instance name SignCheck:

```
Program = attman5.exe
Group = SignCheck
Port = 2001
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
```

```
StopTrigger = hh:mm  
PipeTimeout = 60  
ServerPipeRestartTime = 30  
WriteTimeout = 30  
ServerTimeout = 60  
ShowMode = 0
```

Archive Interface Server

The following private configuration parameters can be set for the Archive Interface Server using the instance type ArchiveServer:

```
Program = SrvMInitJava.exe  
ClassPath =  
    archive.jar;ams.jar;amsDefines.jar;liveinfo.jar;custom.zip;spclien  
    t.jar;spnative.jar;sputils.jar;tiff.jar;softpro.jar;ServerInterfac  
    e.jar;XjLog-1.0.9.jar;logback-classic-1.1.2.jar;logback-core-  
    1.1.2.jar;slf4j-api-  
    1.7.7.jar;xercesImpl.jar;xmlParserAPIs.jar;jdom.jar;jh.jar;db2jcc.  
    jar;db2jcc_license_cu.jar;.;  
MainClass = Archive  
Group = ArchiveIF  
Port = 2015  
ShowMode = 0  
JavaDLL = <JVM dll pathname>
```

Workflow Router

The following configuration parameters can be set for the Workflow Router using the WorkFlowRouter instance type:

```
Program = WFRouter.exe  
Group = Workflow  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
StartTime = hh:mm  
StopTime = hh:mm  
ShowMode = 0  
Userid =  
Password =  
Database =  
Port = 2014  
WFPort = 2018  
CRSPort = 2019  
ConfigFile = .\WFRouter.ini  
CRSRuleFile = crs.xml  
CRSDtdFile = crs.dtd
```

CRS Engine

The following configuration parameters can be set for the CRS Engine using the CRSServer instance type:

```
Program = CRSSrv.exe  
Group = CRS  
StartTrigger = hh:mm  
StopTrigger = hh:mm
```

```
StartTime = hh:mm  
StopTime = hh:mm  
ShowMode = 0  
Userid =  
Password =  
Database =  
CRSConfigFile = crs.ini
```

Schedule Server [SchedSrv]

The following configuration parameters can be set for the Schedule Server using the SchedulerServer instance type:

```
Program = SchedSrv.exe  
Group = Schedule  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
StartTime = hh:mm  
StopTime = hh:mm  
ShowMode = 0  
SchedPort = 2022  
SchedXmlFile = schedule.xml  
SchedXmlWorkFile = schedule.wrk  
SchedXmlErrFile = schedule.err  
SchedDtdFile = SPSchedule.dtd
```

Schedule clients use:

```
SchedCache = schedule.xml  
SchedDtdFile = SPSchedule.dtd  
SchedPort = 2022  
SchedServer = localhost
```

ASV Automat Attach Manager

The following configuration parameters can be set for the ASV Automat Attach Manager using the AttachManager instance type and Automat instance name:

```
Program = attman5.exe  
Group = Automat  
Port = 2002  
StartTime = hh:mm  
StopTime = hh:mm  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
PipeTimeout = 60  
ServerPipeRestartTime = 30  
WriteTimeout = 30  
ServerTimeout = 60  
ShowMode = 0
```

ASV Automat Server

The following configuration parameters can be set for the ASV Automat Server using the instance type ASVServer:

```
Program = scs2.exe
Group = Automat
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
ServerTimeout = 40
ShowMode = 0
Userid =
Password =
Database =
DatabaseSC =
DelayOpenDB = No
SignBase = U
SignCheck = Yes
NoHistory =
SCSnippetPrio =
SCSnippetPrioCheque =
SCSnippetPrioGiro =
Latin = 1,7
VirtualBufferLimit =
VirtualBufferStart =
MsgTypeList = 80,82,8A,8C,A9
LogEngineMatchRate = N
WriteStatistics = Yes
ARVResultFeatureID = 28
ARVQueue = 28
```

APSV Attach Manager

The following configuration parameters can be set for the APSV Attach Manager using the instance type AttachManager and instance name APSV:

```
Program = attman5.exe
Group = APSV
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
PipeTimeout = 0
ServerPipeRestartTime = 30
WriteTimeout = 30
ServerTimeout = 0
ShowMode = 0
```

APSV Server

The following configuration parameters can be set for the APSV Server with the instance type APSVServer:

```
Program = scp2.exe
Group = APSV
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
ShowMode = 0
Userid = signplus
Password = signplus
```

```
DBServer =  
Database = signplus  
DatabaseSC =  
DelayOpenDB = No  
SignBase = U  
SignCheck = No  
VirtualBufferLimit = 67108864  
VirtualBufferStart = 8192  
MsgTypeList = 32  
NoHistory = No  
Sca2Ex = N  
APSVBase = STVQ  
MasterWait = -1  
ResultCycle = 1000  
RequestCycle = 1000  
WaitLimit = 20000  
ServerTimeout = 0  
PipeTimeout = 0
```

APSV (getNext) Server

The following configuration parameters can be set for the APSV Server with the instance type APSVServer:

```
Program = scp2.exe  
Group = APSV  
StartTime = hh:mm  
StopTime = hh:mm  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
ShowMode = 0  
Userid = signplus  
Password = signplus  
DBServer =  
Database = signplus  
DatabaseSC =  
DelayOpenDB = No  
SignBase = U  
SignCheck = No  
VirtualBufferLimit = 67108864  
VirtualBufferStart = 8192  
MsgTypeList = 80,34,35  
NoHistory = No  
Sca2Ex = N  
APSVBase = STVQ  
MasterWait = -1  
ResultCycle = 1000  
RequestCycle = 1000  
WaitLimit = 20000  
ServerTimeout = 0  
PipeTimeout = 0
```

SDQ Attach Manager

The following configuration parameters can be set for the SDQ Attach Manager using the instance type AttachManager and instance name SDQ:

```
Program = attman5.exe  
Group = SDQ
```

```
StartTime = hh:mm  
StopTime = hh:mm  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
PipeTimeout = 0  
ServerPipeRestartTime = 30  
WriteTimeout = 30  
ServerTimeout = 0  
ShowMode = 0
```

SDQ Server

Only one SDQ server may be configured on any machine. The following configuration parameters can be set for the SDQ Server using the instance type SDQServer:

```
Program = stl.exe  
Group = SDQ  
StartTime = hh:mm  
StopTime = hh:mm  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
ShowMode = 0  
Userid =  
Password =  
Database =  
VirtualBufferLimit = 102400  
DelayOpenDB = No  
SignBase = None  
MsgTypeList = 30,37  
APSVBase = STVQ  
ICVServerFactor = 1  
LoopDelayCycleTime = 3600000  
ServiceThreadStack = 200  
ReplyPipeSleep = 10  
ReplyPipeWait = 30000  
InsertHistorised = N
```

ICV Server

The following configuration parameters can be set for the ICV Server with the instance type ICVServer:

```
Program = svl.exe  
Group = ICV  
StartTime = hh:mm  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
ShowMode = 0  
Userid = signplus  
Password = signplus  
Database = signplus  
PasswordSC = signplus  
DatabaseSC = signplus  
VirtualBufferLimit = 102400  
DelayOpenDB = No  
SignCheck = Yes  
APSVBase = STVQ  
ICVServerFactor = 1
```



```
LoopDelayCycleTime = 3600000
ServiceThreadStack = 200
ReplyPipeSleep = 10
ReplyPipeWait = 30000
ICVGetterFeatureId = 9997
ICVPutterFeatureId = 9998
ICVGetterQueue = 97
ICVPutterQueue = 98
NotifyTimeout = 12000
CRSRouterPort = 2020
CRSRouterAddress = localhost
ICVPutterUser = ICVPutter
ICVDocPrefix = ICV
```

GIA Attach Manager

The following configuration parameters can be set for the GIA Attach Manager using the instance type `AttachManager` and instance name `GIA`:

```
Program = attman5.exe
Group = GIA
Port = 2021
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
PipeTimeout = 60
ServerPipeRestartTime = 30
WriteTimeout = 30
ServerTimeout = 60
ShowMode = 0
```

GIA Server

The following configuration parameters can be set for the GIA Server using the instance type `GIAServer`:

```
Program = scg2.exe
Group = Automat
StartTime = hh:mm
StopTime = hh:mm
StartTrigger = hh:mm
StopTrigger = hh:mm
ServerTimeout = 40
ShowMode = 0
Userid =
Password =
Database =
DatabaseSC =
DelayOpenDB = No
SignBase = U
SignCheck = Yes
NoHistory =
Latin = 1,7
VirtualBufferLimit =
VirtualBufferStart =
MsgTypeList = 80,82,8A,8C,A9
LogEngineMatchRate = N
WriteStatistics = Yes
```

[AutomatQueues](#) = 25

Service Programs

The following configuration parameters can be set for the Service Programs:

```
MainClass =  
ClassPath =  
StartTime = hh:mm  
StopTime = hh:mm  
JavaDLL = <JVM dll pathname>  
JavaOpts =  
ApplArgs =  
ALDongle = No  
StartTrigger = hh:mm  
StopTrigger = hh:mm  
Group =  
ShowMode = 0
```

FraudOne Administration Client

General

The following chapter details the FraudOne Administration Client standard functionality. The availability of certain Clients and functionalities in your installation may vary depending on the products you have purchased.

FraudOne Administration Client standard functionality

The FraudOne Administration Client provide following functionalities:

- System Configuration
- User Administration
- CRS (Combined Risk Score) Editor
- Blacklist Administration
- License Report

FraudOne Administration Client configuration

The FraudOne functionality allows for configuration of the Administration Client by using configuration properties files. These files, located in the FraudOne SignBase database, allow the user to specify the following standard settings:

- Connection to the SignBase and SignCheck servers
- Connection settings for the Clients
- Look and feel of the Clients
- Security settings

FraudOne Administration Client requirements

The following are the requirements for the Administration Client:

- Requires an installed Java Runtime Environment. See *Kofax FraudOne Installation and Migration Guide*.
- A valid TCP/IP connection to the server hosting the SignBase and/or SignCheck Application Servers.
- A minimum screen resolution of 1024x768 on the workstation where the Client is used.

Administrating with the Application Server configuration file elements

Using the configuration file elements for the SignBase application server, located in the SignBase database, enables for the following:

- Define the number of unsuccessful login attempts allowed before a user is blocked
- Define the parameters used for re-introducing a previously used password
- Enable message encryption between FraudOne Clients and the connected servers. Contact your Kofax business representative for further details.
- Specify the threshold of requested items being sent to the Administration Client.

Refer to the [Server Manager](#) chapter which describes administration and configuration using these configuration file elements. The configuration so specified applies to all the FraudOne Clients connected to the Server Manager.

The FraudOne Administration Client can be configured by using its corresponding configuration properties files which are also located in the FraudOne SignBase database.

Administration Client

The Administration Client (Admin Client) provides Graphical User Interface functionality for administrative work in the FraudOne system.

The Administration Client allows the following when connected to the SignBase and SignCheck databases:

- Designate and edit new users and groups with the **User Administration**.
- View and maintain PayeeName Blacklist, PayeeVerification list and Checkstock Blacklist with the **Blacklist Administration**
- Create and edit rule graphs for CRS with the **CRS Editor**
- View and maintain topography records and configuration file elements with the **System Configuration**
- Generate **Licence Reports** in which all licence relevant data is listed

Installing the Administration Client

The standalone Administration Client is included in the FraudOne core functionality. In order to install the Administration Client to a connected workstation simply copy the Administration Client directory to the workstation.

Important Refer to the *Kofax FraudOne Installation and Migration Guide* for further installation considerations.

Installing the Administration Client icon

The Administration Client can be started by double-clicking the **AdminClient.cmd** command file located in the SignPlus directory. Optionally, a shortcut to the command file can be created.



The standard FraudOne installation includes the Administration Client icon named **signplus_n.ico**, which can be installed on the workstation. In order to install the icon do the following:

1. Create a shortcut to the Administration Client. Right click the location where the shortcut is to be placed, then select
New > Shortcut
2. In the next dialog box that appears write the Administration Client command file **AdminClient.cmd** location. Optionally, click the **Browse** button to locate the file.

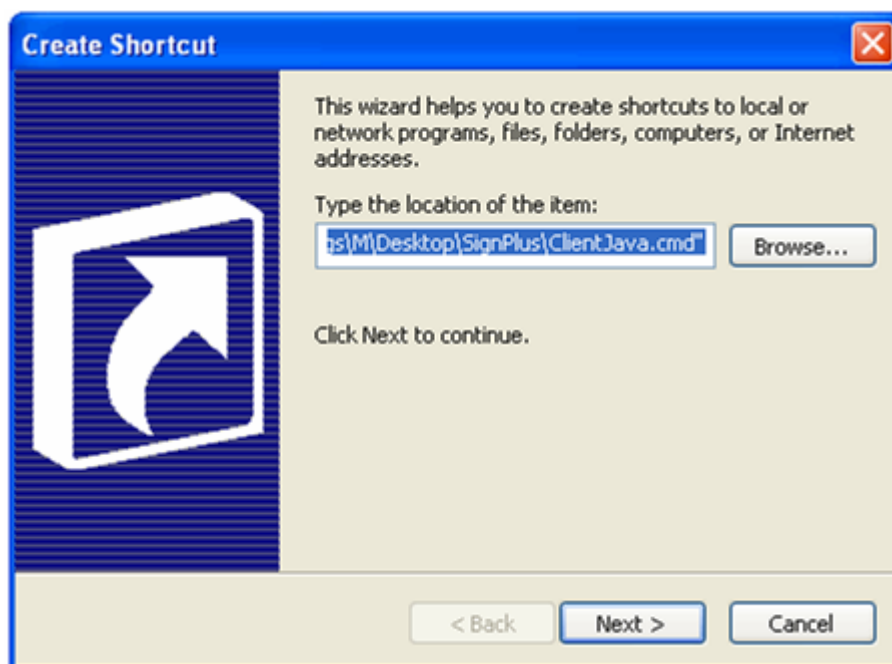


Figure 8: Create shortcut dialog box

3. Click **Next**. In the next dialog box, type the name for the shortcut and click **Finish**. This creates



the Administration Client shortcut in the specified location.

4. In order to change the icon, right-click the newly created shortcut and select **Properties**. In the following dialog box, click the **Change Icon** button.

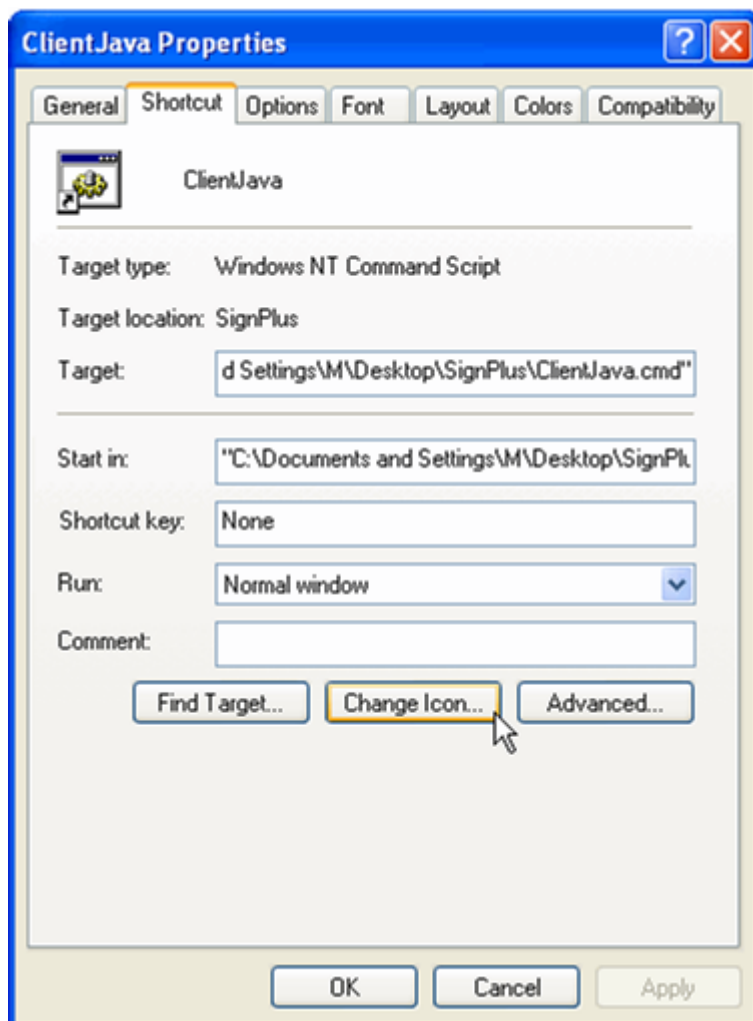



Figure 9: ClientJava Properties

5. Click **OK** on the next warning message box.
6. This brings up the following dialog box:



Figure 10: Choosing an icon

7. Click **Browse** to select the Administration Client icon  located in the SignPlus directory. Optionally, you may choose an icon from the ones provided. When the desired icon appears in the selected icon list click **OK**. Lastly, click **OK** in the **Properties** dialog box in order to accept the changes.

Initial setup

After installation and before the first invocation the Administration Client is supposed to be setup with **SetupCfg.exe** utility from the distributive. In particular, a configuration server and port are specified during the installation. See the *Kofax FraudOne Installation and Migration Guide* for more details.

Configuring the Administration Client

This section details the configuration possibilities with the properties files. All property files are stored in the Central Configuration database and can be accessed and managed via the System Configuration panel of the Administration Client, for more details read the *FraudOne Administration Client Online Help*.

The **admin.properties** file, located in the Central Configuration database, is the main configuration file for the FraudOne Administration Client. Once the Client is installed on the workstation the properties file can be configured from the System Configuration panel of the Administration Client.

Configuration of the property file allows the specifying of the following:

- Server connection settings

- Trace levels settings
- Look and feel of the Client

The `admin.properties` configuration file element

The **admin.properties** configuration file element is the main configuration file of the Administration Client. The following tables describe the different configurations available.

Important Entries within this file have precedence over entries having the same name in other resource files.

Every entry is preceded by a prefix in order to determine the configuration and which resource files an entry belongs to.

Example

```
SignCheck.Port = 2001
```

This defines the port number for communication with the SignCheck server where the default is 2001.

Connection parameters

The following table describes the configuration settings of the Administration Client connection settings:

Parameter	Description
<code>SignBase.Host =</code>	IP address of the SignBase server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>SignBase.Port = 2000</code>	Port number for communications with the SignBase server. Must be in sync with the port number specified in the [SignBase] section of the SrvMgr4.ini configuration file. Default: 2000
<code>SignBase.Timeout = 30000</code>	Maximum time (in milliseconds) for a request to wait for SignBase server to answer. Default: 30000
<code>ConfigServer.Host =</code>	IP address of the configuration server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>ConfigServer.Port =</code>	Port number for communications with the configuration server. Must be in sync with the port number specified in [SignBase] section of the SrvMgr4.ini configuration file.

Parameter	Description
	Default: 2000
<code>ConfigServer.Timeout =</code>	Maximum time (in milliseconds) for a request to wait for configuration server to answer. Default: 60100
<code>SignCheck.Host =</code>	IP address of the SignBase server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>SignCheck.Port = 2001</code>	Port number for communications with the SignCheck server. Must be in sync with the port number specified in [SignCheck] section of the SrvMgr4.ini configuration file. Default: 2001
<code>SignCheck.Timeout = 30000</code>	Maximum time (in milliseconds) that a request waits until the SignCheck server answers. Default: 30000
<code>SignCheck.Enabled =</code>	0 - SignCheck not used. SignCheck will not make requests to the SignCheck server and SignCheck menus will not be displayed in the menu bar. 1 - A request to the specified SignCheck server will be made to load the SignCheck queues and to display the SignCheck menu items. If no SignCheck server can be found, an error message will be displayed. Default: 1
<code>SignCheck.NotifyHost =</code>	Name of server where the WFRouter is located.
<code>SignCheck.NotifyPort = 2014</code>	Port number for communications with the SignCheck workflow router. Must be in sync with the port number specified in the Workflow section [WFRouter] of the SrvMgr4.ini configuration file. Default: 2014

Trace level

The following table describes the configuration of the Administration Client trace levels settings:

Parameter	Description
<code>TraceLevel = 1</code>	The trace level is a combination (bitwise OR) of the following values:

Parameter	Description
	-1 - Trace everything 0 - No trace at all 1 - Error messages 2 - Warnings 4 - Debugging information 8 - Information 16 - Resource information 32 - SQL Trace 64 - Substitute (internal usage only) 128 - Selection (internal usage only) 256 - Performance Important Default is minimum TraceLevel = 1
TraceFile = logs/adminClient.log	Path and name of the trace file. If not specified, trace information will not be written onto your hard disk.

Default settings

The following table describes the configuration of the Administration Client default settings:

Parameter	Description
UseSignSec = 1	0 - FraudOne security is turned off 1 - FraudOne security is turned on Default: 1

Property file location

The following table describes the configuration of the Administration Client property file location:

Parameter	Description
Globals.Resources.Path =	The core path for Administration Client properties has to be set to de/softpro/adminclient/resources/
Custom.Resources.Path =	Customer specific path for Administration Client properties Example de/demo/adminclient/resources/

BNO specific settings

For the Administration Client the following table describes the configuration of the BNO settings:

Parameter	Description
BNO.List = 999	Available BNOs 999 is reserved and means "all BNOs".

User Administration settings

The following parameters describe the configuration of the **User Administration** settings for the Administration Client:

```
UserAdmin.Visible = 1
UseDefaultBrowser = 1
HTMLBrowser.Path =
UserAdmin.QuickAccess.Visible = 0
```

Parameter Administration settings

The following parameters describe the configuration of the **Parameter Administration** settings for the Administration Client:

```
ParamAdmin.Visible = 1
ParamAdmin.UseExtended = 0
ParamAdmin.DisplayCRSEntries = 1
```

CRS Editor settings

The following parameters describe the configuration of the CRS editor settings for the Administration Client:

```
CRS.Visible = 1
CRS.DirectImport.Visible = 0
CRS.DirectUpgrade.Visible = 0
CRS.LoadActiveRule = 1
CRS_Editor.RuleBlocks.Enabled = 1
```

Blacklist Administration settings

The following parameters describe the configuration of the blacklist administration settings:

```
Blacklist.Visible = 1
Blacklist.PayeeName.Visible = 1
Blacklist.PayeeVerification.Visible = 1
Blacklist.StockImage.Visible = 1
Blacklist.GlobalFraudSignatureList.Visible = 1
```

Limitations of the Administration Client

When using the FraudOne Administration Client the following should be considered:

- Only one instance of Administration Client can be operated per workstation

FraudOne clients

General

The following chapter details the FraudOne Clients in the standard functionality as well as the configurations that can be performed by an administrator. The availability of certain Clients and functionalities in your installation may vary depending on the products you have purchased.

Overview FraudOne clients

The FraudOne Clients allow for a visual interface to be used in the FraudOne environment. The Clients provide the user with a means to view and to perform the general maintenance of accounts and customers. The core Client components are comprised of the following interfaces:

1. FraudOne Java Client: a stand-alone GUI that is used for maintenance of customers, accounts and its signatories in the SignBase and SignCheck databases. The Java Client can also be used for the administration of users settings
2. FraudOne Teller Interface: allows existing front office systems to interface with the Java Client and Thin Client.
3. FraudOne Thin Client: a web based GUI Client that is connected over a company's intranet and provides SignBase 'display-only' functionality. The Thin Client also provides full SignCheck functionality by allowing the end user to accept or reject checks.

FraudOne clients standard functionality

The FraudOne Clients provide various functionalities when connected to the following databases:

When connected to the SignBase database (Default functionality) the user can:

- Display reference data. This includes customer, account, signatories and rules. This functionality is available with the Java Client and Thin Client.
- Perform reference maintenance which includes:
 - Importing images via a scanner connected to the workstation. This functionality is only available with the Java Client when it is connected to the SignInfo database
 - Perform reference maintenance which includes the capturing of signatures via a signature pad connected to the workstation. This functionality is only available with the Java and Thin Clients.
- Administration of user and access rights within the FraudOne system. This functionality is only available with the Java Client.
- Auditing feature which allows the retrieving of customer or account information at a specific time/date for comparison purposes. This functionality is only available with the Java Client.

When connected to the SignCheck database the user is able to do the following:

- Monitor the SignCheck workflow and processing status of the following verification engines:
 - Automatic Signature Verification (ASV)
 - Check Stock Verification (CSV)
- Carry out visual verification on checks in a queue
- Carry out physical verification on checks in a queue
- Retrieve the lists for queues, user decisions/transactions, and search lists
- Show all decisions and statistics made for a check

The diagram below describes the standard FraudOne Client functionality when connected to the FraudOne databases:

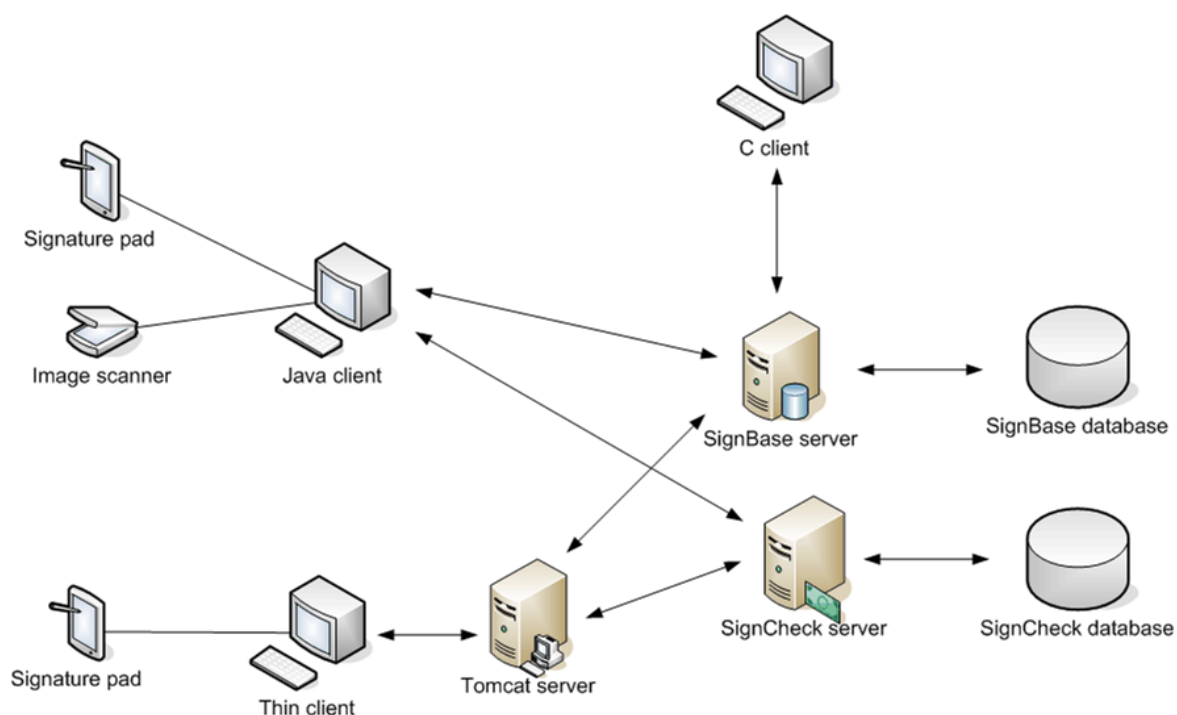


Figure 11: FraudOne Clients communication

FraudOne client configuration

The FraudOne functionality allows for configuration of the Java Client and Thin Client by using configuration files. The property files are maintained in the Central Configuration and allow the user to specify the following standard settings:

- Connection to the SignBase and SignCheck servers
- Connection settings for the Clients
- Trace levels for the logs

- Look and feel of the Clients
- Security settings

Other available configuration settings depend on the FraudOne Client used. The configuration possibilities are described in each of the respective Client sections.

FraudOne client requirements

The following are the requirements for the Java Client and Thin Client:

- Requires an installed Java Runtime Environment for the Java and Thin Clients only. See *Kofax FraudOne Installation and Migration Guide*.
- A valid TCP/IP connection to the server hosting the SignBase and/or SignCheck Application Servers.
- Installation of CFM Twain driver required for scanning features. See *Kofax FraudOne Installation and Migration Guide*.
- Installation of appropriate pad drivers for live signature capture. See *Kofax FraudOne Installation and Migration Guide*.
- A minimum screen resolution of 1024x768 on the workstation where the Client is used.

Administrating with the SrvMgr4.ini configuration file

The Server Manager, which is included in the FraudOne core functionality, is used as a control interface for the different FraudOne servers. The **SrvMgr4.ini** configuration file can specify certain parameters of the Clients. It is stored on the server and can be accessed and modified with help of the Administration Client. The configuration file enables for the following:

- Define the number of unsuccessful login attempts allowed before a user is blocked.
- Define the parameters used for re-introducing a previously used password.
- Enable message encryption between FraudOne Clients and the connected servers. Contact your Kofax business representative for further details.
- Specify the threshold of requested items being sent to the FraudOne Clients. This feature applies to Java and Thin Clients only.

Refer to the Server Manager chapter which describes administration and configuration using the **SrvMgr4.ini** file. The configuration in this file applies to all the FraudOne Clients connected to the Server Manager.

The FraudOne Clients (Java Client and Thin Client) can be configured by using their corresponding configuration files which are maintained in the Central Configuration. Refer to the specific FraudOne Client sections for detailed descriptions on using the Client specific configuration file.

FraudOne Java Client

The Java Client provides Graphical User Interface functionality for the use of physical and automatic verification of checks, images, and signatories. The Java Client accesses the SignBase database for reference maintenance or accesses the SignCheck database to perform visual verification.

The Java Client allows for the following when connected to the SignBase and SignCheck databases:

- View and edit the customer's accounts, signatories, signatures, and rules
- Designate and edit new users
- Import data, images and XML files from the Archive Interface Server (SignBase database)
- Audit Search and Compare function which keeps logs of user transactions (SignBase database)
- Scan Control feature for scanning signatures from a scanner or signature pad (SignBase)
- Physical verification of checks, signatures, and check stock (SignCheck)
- Live verification of a signature signed on a pen-pad (SignBase)

Installing the Java Client

The standalone Java Client is included in the FraudOne core functionality. In order to install the Java Client to a connected workstation simply copy the Java Client directory to the workstation.

Note The FraudOne installation allows for multiple configuration files to be installed in one directory. This allows for several different configurations to be available for the end users. It is also possible to have several **custom.properties** files with different extensions but important to name the original one **custom.properties** file. As all other configuration files, the **custom.properties** is stored in the Central Configuration.

See *Kofax FraudOne Installation and Migration Guide* for further installation considerations.

Installing the Java Client icon

The Java Client can be started by double-clicking the **ClientJava.cmd** command file located in the SignPlus directory. Optionally, a shortcut to the command file can be created.



The standard FraudOne installation includes the Java Client icon named **signplus_n.ico**, which can be installed on the workstation. In order to install the icon do the following:

1. Create a shortcut to the Java Client: Right click the location where the shortcut is to be placed, then select
New > Shortcut
2. In the next dialog box that appears write the ClientJava command file **ClientJava.cmd** location. Optionally, click the **Browse** button to locate the file.

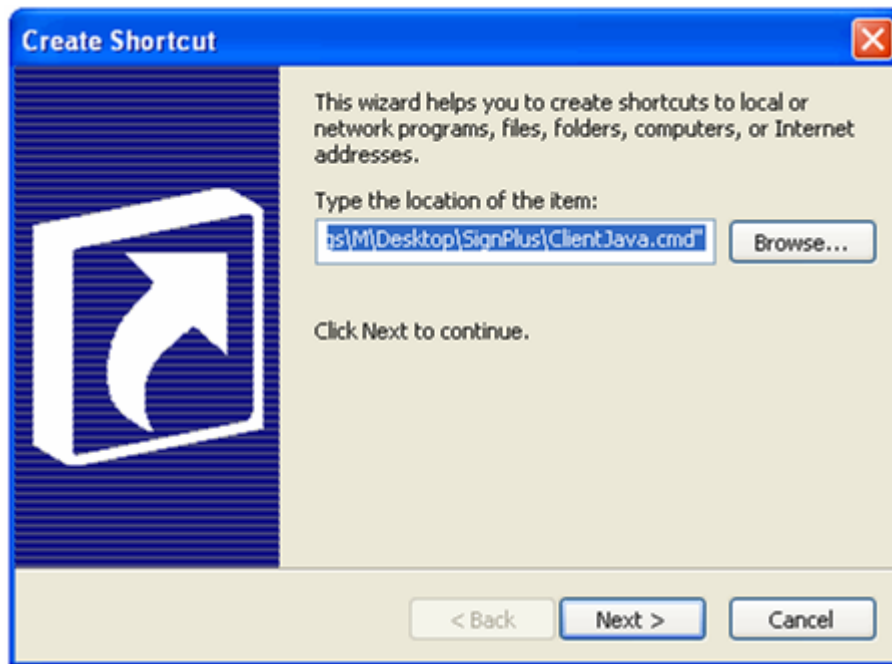


Figure 12: Create shortcut dialog box

3. Click **Next**. In the next dialog box, type the name for the shortcut and click **Finish**. This creates

the Java Client shortcut   in the specified location.

4. In order to change the icon, right-click the newly created shortcut and select **Properties**. In the following dialog box, click the **Change Icon** button.

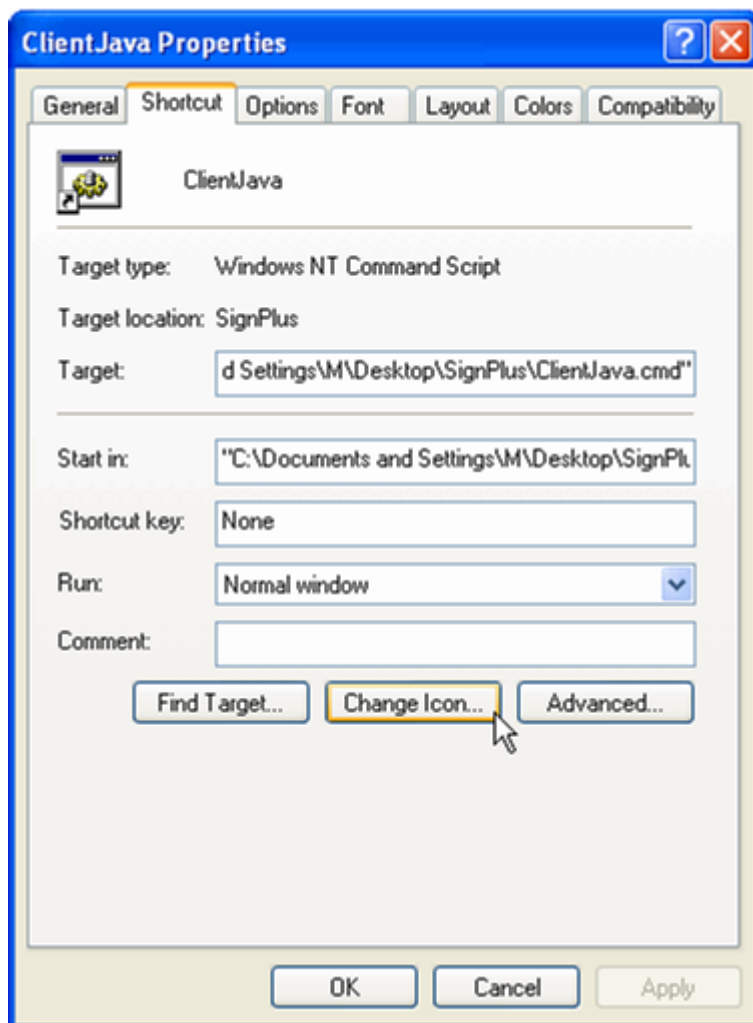



Figure 13: ClientJava Properties

5. Click **OK** on the next warning message box.
6. This brings up the following dialog box



Figure 14: Choosing an icon

7. Click **Browse** to select the Java Client icon  located in the SignPlus directory. Optionally, you may choose an icon from the ones provided. When the desired icon appears in the selected icon list click **OK**. Lastly, click **OK** in the **Properties** dialog box in order to accept the changes.

Initial setup

After installation and before the first invocation the Java Client is supposed to be setup with **SetupCfg.exe** utility from the distributive. In particular, a configuration server and port are specified during the installation. See the *Kofax FraudOne Installation and Migration Guide* for more details.

Using the Java Client

The Java Client may be configured to operate in the following modes:

- **Standard Java Client**
This mode is the default mode that is used for maintenance of customer information with SignBase and for Visual Signature Verification with SignCheck.
- **Teller mode**
Allows integration into an existing system via a Teller interface. This mode can be configured to allow external applications to access FraudOne functionality directly without any user interaction. The [Configuring the Java Client](#) section below describes the configuration of the Java Client using the **custom.properties** file, located in the SignPlus installation directory.

The **TellerInterface.Enabled** parameter is located in the 'Default Settings' table and enables the Teller mode operation.

Scan Control Image import

The FraudOne Scan Control Image import allows the scanning of images, documents and signatures into the SignBase database. Images and documents include copies of account cards (opening account forms), driver's license, passports, and other images that are used for reference purposes. Images can be scanned via a scanner installed on a workstation.

Additionally, the SignBase Java Client includes the added functionality of importing signatures via a signature pad installed on the workstation.

Scanning requirements

The images should be scanned with a minimum resolution of 200 DPI, black & white, or 8-bit gray. Black & white may be the better choice to reduce the resources needed to save images.

The following are the supported image file formats:

- TIFF: Tagged Image File Format, uncompressed;
- BMP: Bitmap File Version 2, uncompressed

Optional fields in the image files should be configured to have the following values:

- Bit depth (black-white: 1 bit per pixel, grey: 8 bit per pixel)
- Dpi resolution

Black and White images should be created with a contrast setting resulting in black signature strokes with a white background. This minimizes the amount of manual cleaning needed afterwards. The contrast in grey images should be reduced in order to prevent color saturation since grey images require a larger amount of storage space in the SignBase database.

Note Several versions of Tagged Image File Formats are currently in use. Some add extended tags, which may be application specific and may then be unreadable. OS2 Bitmaps Version 1.0 is not usable, since it does not allow for dpi resolution.

Scanner driver installation requirements:

- Basic Twain driver should be installed when using Windows NT 4.0 operating system. A basic TWAIN driver for NT4.0 is included in every Kofax delivery as part of the system software. The basic TWAIN driver is installed in Windows 2000 and Windows XP operating systems.
- CFM Twain 7.x must be installed on each scanning workstation. Each scanning workstation requires an individual license. Consult your Kofax business representative for further information regarding licensing.
- If CFM is installed, no additional propriety scanner driver is required.
- CFM does not support the HP scanner models 74xx or higher.

Signature images scanned via a signature pad must meet the following requirements:

- TIFF format only
- Non-loss compression type
- Minimum of 200dpi resolution

Using the Scan Control Image import

The **custom.properties** file allows for the configuration of the Scan Control Image Import settings. Refer to the Scan Control settings table located in the section [The custom.properties configuration file](#).

Using the Java Client in Teller Mode

The Teller mode can be configured to allow external applications to access FraudOne functionality directly without user interaction. The **custom.properties** file detailed in [Configuring the Java Client](#) section contains the **TellerInterface.Enabled** parameter (in the default settings table) which enables the Teller mode.

Help feature

The FraudOne Java Client includes an embedded Help feature that can be accessed by the menu bar. This detailed feature includes explanations on the different terminology as well as common usage of the Java Client in the FraudOne system.

Configuring the Java Client

This section details the configuration possibilities with the properties files. All property files are stored in the Central Configuration database and can be accessed and managed via the System Configuration panel of the Administration Client, for more details see the *Kofax FraudOne Administration Client Help*.

The **custom.properties** file is the main configuration file for the FraudOne Java Client. This file, as all other configuration files, is stored on the central configuration server and can be accessed and modified from the Administration Client.

Configuration of the property file allows the specifying of the following:

- Server connection settings
- Trace levels settings
- Look and feel of the Client
- Scan Control Image Import settings
- Variant settings
- Verification rate settings

The custom.properties configuration file

The **custom.properties** file is the main configuration file of the Java Client. The following describes the different configurations available.

Important Entries within this file have precedence over entries having the same name in other resource files.

Every entry is preceded by a prefix in order to determine the configuration and which resource files an entry belongs to.

Example

```
SignCheck.Port = 2001
```

This defines the port number for communication with the SignCheck server where the default is 2001.

Connection settings

The following table describes the configuration settings of the Java Client connection settings:

Parameter	Description
<code>SignBase.Host</code>	IP address of the SignBase server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>SignBase.Port</code>	Port number for communications with the SignBase server. Must be in sync with the port number specified in the section [SignBase] of the SrvMngr4.ini file. Default: 2000
<code>SignBase.Timeout</code>	Maximum time (in milliseconds) for a request to wait for SignBase server to answer. Default: 30000
<code>SignBase.CopySignatoriesAction.Timeout</code>	Maximum time (in milliseconds) that a request to copy signatories to another customer/account waits until the SignBase server answers. If this property is not set or less or equal than zero the default timeout (SignBase.Timeout) will be used. This property enables different timeout settings to copy signatories. This is necessary when copying a huge amount of signatories. Default: 30000
<code>ConfigServer.Host</code>	IP address of the configuration server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>ConfigServer.Port</code>	Port number for communications with the configuration server. Must be in sync with the port number specified in [SignBase] section of the SrvMngr4.ini file. Default: 2000

Parameter	Description
<code>ConfigServer.Timeout</code>	Maximum time (in milliseconds) for a request to wait for configuration server to answer. Default: 60100
<code>SignCheck.Host</code>	IP address of the SignBase server Default: the host specified as argument to SetupCfg.exe during the client setting up
<code>SignCheck.Port</code>	Port number for communications with the SignCheck server. Must be in sync with the port number specified in [SignCheck] section of the SrvMgr4.ini configuration file. Default: 2001
<code>SignCheck.Timeout</code>	Maximum time (in milliseconds) that a request waits until the SignCheck server answers Default: 30000
<code>SignCheck.Enabled</code>	0 - SignCheck not used. SignCheck will not make requests to the SignCheck server and SignCheck menus will not be displayed in the menu bar. 1 - A request to the specified SignCheck server will be made to load the SignCheck queues and to display the SignCheck menu items. If no SignCheck server can be found, an error message will be displayed. Default: 1
<code>SignCheck.NotifyHost</code>	Name of server where the WFRouter is located.
<code>SignCheck.NotifyPort</code>	Port number for communication with the workflow router. Must be in sync with the port number specified in the Workflow section [WFRouter] of the SrvMgr4.ini configuration file. Default: 2014
<code>STV.Host</code>	IP address of the STV server Default: the SignCheck host
<code>STV.Port</code>	Port number for communication with the STV server Default: 2007
<code>STV.Timeout</code>	Maximum time (in milliseconds) that a request waits until the STV server answers Default: 30000
<code>ArchiveServer.Host</code>	IP address of the Archive Interface Server (AIS)

Parameter	Description
	Default: the SignCheck host
<code>ArchiveServer.Port</code>	Port number for communications with the Archive Interface Server (AIS) Default: 2015
<code>ArchiveServer.Timeout</code>	Maximum time (in milliseconds) that a request waits until the Archive Interface Server (AIS) answers Default: 30000
<code>ICS.Host</code>	IP address of the Inwards Clearing Server This property is only valid if the Java Client has been configured to make use of a central Inwards Clearing Server. Default: localhost
<code>ICS.Port</code>	Port number of the Inwards Clearing Server This property is only valid if the Java Client has been configured to make use of a central Inwards Clearing Server. Default: 1099
<code>DisableSaveAfterServerTimeout</code>	This settings (since release 3.3.20) triggers the behavior of the Java client when saving the changes to the database. 0 - Users are allowed to save the data again after a timeout has occurred. 1 - In the case of a timeout the user will not be able to store the data if the data is already stored. This prevents the user from storing the changes more than once. Default: 0

Trace level

The following table describes the configuration of the Java client trace levels settings:

Parameter	Description
<code>TraceLevel = 1</code>	The trace level is a combination (bitwise OR) of the following values: -1 - Trace everything 0 - No trace at all 1 - Error messages 2 - Warnings

Parameter	Description
	4 - Debugging information 8 - Information 16 - Resource information 32 - SQL Trace 64 - Substitute (internal usage only) 128 - Selection (internal usage only) 256 - Performance Note Default is minimum TraceLevel = 1
TraceFile = logs/jclient.log	Path and name of the trace file. If not specified, trace information will not be written onto your hard disk.

Default settings

The following table describes the configuration of the Java Client default settings:

Parameter	Description
BankNo	For multiple-client capability: Standard Bank number entry in search dialogs. BankNo = 999 is not permitted
CountryId	Country identification, refers to country telephone code. Default: 049
BankCode	Standard BankCode (such as routing code) entry in search dialogs. Default: blank
Currency	Standard currency used Default: USD
LimitSignature	Maximum number of authorized signatories for whom signature graphics will be transmitted directly. Default: 10
UseSignSec	0 - FraudOne security is turned off 1 - FraudOne security is turned on Default: 1
UseChecksum	0 - no checksum calculation

Parameter	Description
	1 - checksum calculation for customer and/or account (bank-specific configurable) Default: 0
UseTextIcons	0 - eye-catchers are displayed as symbols 1 - eye-catcher are displayed as text Default: 1
ShowSignatoryNumbers	Displays the signatory numbers in the list overview. Default: 1
HideNoRightSignatories	Since release 3.5.0 0 - signatories are displayed independently of mandates 1 - signatories with no rights will not be displayed in the signatory list 2 - signatories with no rights will not be displayed in the signatory list if the user does not have update rights Default: 0
HideVariants	Since release 3.5.0 0 - variants are displayed 1 - variants are not displayed in signatory list Default: 0
HideNotValidVariants	Since release 3.5.16 0 - variants are displayed ignoring the variants valid from date 1 - variants not at query date valid are not displayed in signatory list Default: 0
EnableVariantToReferenceAssignment	Since release 3.5.22 0 - replacement of reference signature is not possible 1 - in variant assignment dialog users are also able to replace the signatories reference signature with the signature of a chosen variant Default: 0
TellerInterface.Enabled	Since release 3.5.0 0 - Teller API is not started 1 - the client starts API for Teller access) Default: 1
SignInfo	Mode of the SignBase client

Parameter	Description
	0 - run in SignBase mode 1 - run in SignInfo mode Default: 0
Checkstock	Mode of the SignBase client 0 - run in SignBase mode 1 - run in Checkstock mode Default: 0
HideNotValidCheckstocks	0 - checkstocks are displayed ignoring the valid from date 1 - checkstocks not valid at query date are not displayed Default: 0

Look and feel

The following parameters describe the configuration of the Java Client Look and Feel settings:

[FocusColor](#) = yellow
[ShowVerifyIcon](#) = 1

Verification rate settings

The following parameters describe the configuration of the Java Client verification rate settings. The verification rate is entered in per cent.

100 - means that every change to a customer must be reviewed by a second person.

0 - means that a second verification is not required.

The default value for all Verify entries is 100.

[Pverify](#)
[UIVerify](#)
[Overify](#)

Property file location

The following table describes the configuration of the Java Client property file location:

Parameter	Description
Globals.Resources.Path	Specification per customer (bank) Example de/demo/resources/

Scan control settings

The following parameters describe the configuration of the Scan Control Image Import settings:

[ScanCtrl.EditResult](#)
[ScanCtrl.GrayBitmapSizeInt](#)
[ScanCtrl.BWBitmapSizeInt](#)
[ScanCtrl.CheckSizeInt](#)
[ScanCtrl.ResolutionErrAction](#)
[ScanCtrl.TwainDriversInstalled](#)
[ScanCtrl.PadDriver](#)
[ScanCtrl.PadWidth](#)
[ScanCtrl.PadHeight](#)
[ScanCtrl.PadUseLCD.bool](#)

Queue simulation settings

The following Parameters describe the configuration of the Queue Emulation settings:

[QueueEmulation.ICFilePath](#)
[QueueEmulation.ICFileName](#)

BNO specific settings

The following table describes the configuration of the BNO settings:

Parameter	Description
BNO.List = 999	Available BNOs 999 is reserved and means all BNOs
VarValidFrom.999 = 30	Duration (in days) after a variant becomes valid (calculated from the creation day of the variant). In this case variants for all BNOs will be used after 30 days. Default: 60
CheckstockValidFrom.999 = 30	Duration (in days) after a stock image becomes valid (calculated from the creation day of the stock image). In this case stock images for all BNOs will be used after 30 days. Default: 60

Variants settings

The following parameters describe the configuration of the Variants settings. The numbers entered refer to the maximum number of variants multiplied with the number of active reference signatories.

For example: maxVariants = 10, if the active reference signatories for a given customer is 5 then the maximum number of variants for this customer is 5*10=50.

[autoAssign](#)
[autoAssignPrivate](#)
[autoAssignCorporate](#)
[autoAssignOther](#)
[maxVariants](#)
[maxVariantsAutoAssign](#)
[maxVariantsPrivate](#)
[maxVariantsCorporate](#)
[maxVariantsOther](#)
[maxVariantsAutoAssignPrivate](#)
[maxVariantsAutoAssignCorporate](#)
[maxVariantsAutoAssignOther](#)
[HideUnboundVariants](#)

Image Loader settings

The following parameters describe the configuration of the image loader:

[ImageStack.Root.Path](#)
[ImageStack.Suffix](#)

Limitations of the Java Client

When using the FraudOne Java Client the following should be considered:

- Scanning with the Scan Control Image Import requires a large amount of resources from the workstation used
- Only one instance of FraudOne Java Client can be operated per workstation

Teller Interface

The FraudOne Clients can be integrated into existing front office information system by means of the FraudOne Teller interface. It allows users to query info from the SignBase database using the Teller interface. The Teller interface calls up the FraudOne Client in order to display customer and signature data at the workstation. The Teller interface can also be utilized to invoke Sign Teller Verification (STV) in the FraudOne Java or Thin Client and to pass data in XML format to the Java Client. Digital hand written signature can be verified visually or automatically via the STV.

There are two external APIs available with the Teller interface:

- C-API also known as compatibility interface is a programmed interface.
- The second API is a servlet-based interface. The servlet interface can be used within another web application by calling URLs that match the servlet interface specification.

Refer to the [Related documentation](#) chapter for access to available Kofax documentation.

Installation

The Teller interface comes installed in the FraudOne core functionality. See the *Kofax FraudOne Installation and Migration Guide* regarding proper installation procedures.

Usage

The Teller interface provides access to the SignBase database via the FraudOne Clients. The end user can display, create or edit accounts and customers.

When initiating the Teller interface via an existing front office application, the API is invoked and the requested Client is displayed in the foreground until no longer needed. The Thin Client's web based functionality allows it to be integrated into another browser application.

Configuration

The FraudOne Java Client and Thin Client can be configured to turn on the Teller interface. Refer to the specific Client's configuration section for more information.

Limitations

There can be no more than one instance of Teller interface per machine.

FraudOne Thin Client

The FraudOne Thin Client is a web application initiated by a web server on the network. It is accessed via the web browser Internet Explorer from Microsoft. The Thin Client enables the usage of FraudOne throughout a company's network using standard Intranet and Internet technologies.

The following actions can be performed with the FraudOne Thin Client:

- Display of SignBase, SignInfo and CheckStock data (Images, Signatories, Accounts, Customers, etc.)
- Search for customer and account and displaying the various search results. This includes the customer, account, signatories, rules, and group information
- Can run in SignBase-mode or together with SignInfo-mode and/or CheckStock-mode
- Can run in SignInfo-only-mode (SignInfo Classic Edition)
- Can run in SignCheck-mode with storing and assigning Variants to Signatories, and visually clipping new check image regions to store as new Variants. See the *Kofax FraudOne Installation and Migration Guide* regarding clipping requirements.
- Live verification of a signature signed on a connected pen-pad
- Sign Teller Verification for online verification of signatures and rules
- Integration into an existing system via a Teller interface
- Activating/Deactivating of logging information

Installing the Thin Client

The Thin Client generates dynamic content based on the type of request sent by the web browser. For this reason, it requires an installed servlet engine on the web server. The following Server-side pre-requisites should be considered in a Thin Client installation:

- Should be installed on a separate system.
- Since Java Server Pages are delivered uncompiled, the server system is required to have a Java JDK in the system path.
- A servlet container must be installed on the server (currently supported is Jakarta Tomcat).
- Complete installation is contained within a standard J2EE Web Archive File (**SignPlus.war**). The property files for the Thin Client are maintained in the Central Configuration.
 - See section [Thin Client Property Files](#) for information regarding the property files used for the Thin Client.
 - See section [Business Model Property Files](#) for information regarding the property files used for Business Model required by the Thin Client.
 - All property files must be assigned to the 'InstanceType' - ThinClient.

Note All components required by the Thin Client must be locally available on the server machine. Do not install the components on a network share to be accessed by the server. This results in problems if network communications have been interrupted.

The following Client-side pre-requisites should be considered in a Thin Client Installation:

- Plug-ins and ActiveX downloading must be enabled in the browser.
- JavaScript must be enabled in the browsers.
- A detailed description of the supported browsers and versions can be found in the new release documentation. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

The browser must be setup to retrieve a new version of the page at every visit. This allows for the most up to date content to be displayed. To enable this, select **Tools > Internet Options** from the browser window, in the **Temporary Internet Files** box select **Settings**, and then click **Every visit to the page** radio button.

The installation of the Thin Client requires up to three different components to be installed:

1. Apache Web Server (optional)
2. Jakarta Tomcat servlet container (required)
3. Installation of Thin Client components (required)

The following section describes the installation procedures required for the Thin Client environment. The Installation description of Apache and Tomcat should only be seen here as a small help for configuration, it includes no guaranty to correctness and completeness. For a detailed installation instruction refer to the Apache and Tomcat Jakarta Project documentation. See the *Kofax FraudOne Installation and Migration Guide* document regarding browser installation requirements.

Installation of Apache (Optional)

This step is only necessary if the Apache Web Server is to be used. Apache is provided as a Microsoft Installer Package. Follow the displayed instructions during the installation process.

The integration of Tomcat into Apache Web Server is described in the Apache Tomcat Connector "How To" section. The implementation of Load Balancing or secure HTTPS connection via Apache Web Server is not included in this documentation. It has to be set up independent from Thin Client installation.

Installation of Tomcat

Tomcat is provided as an archive called **apache-tomcat-8.x.xx.zip**. The installation consists of extracting the ZIP file to the web-server-system. In order to provide Tomcat functionalities, the environment-variable CATALINA_HOME must be set and must point to the installation-directory of Tomcat.

Installation of the Thin Client components

The Thin Client components are packaged in a single J2EE Web Application Archive file (.war). War files are a standard method for deploying web applications. The archive contains all components necessary for the successful operation of the web application in a single compact package.

The Thin Client's archive is called **SignPlus.war** by default. To deploy the archive to the Tomcat installation, follow this procedure:

1. Copy the **SignPlus.war** file to the folder:

```
%CATALINA_HOME%\webapps
```

2. Add the following entry to the Windows PATH environment variable:

```
%CATALINA_HOME%\webapps\SignPlus\WEB-INF\lib\bin
```

or copy the file

```
/SignPlus/samples/setenv.bat.sample
```

as **setenv.bat** file to the bin directory of your tomcat installation. **Setenv.bat** is called during the start of the servlet container only if Tomcat is started via **startup.cmd** (and not as Windows service). The possibility to start Tomcat from command line is only provided in the Tomcat archive version (download of apache-tomcat-8.x.xx.zip, whereas x.xx describes the minor version identifier of the container). Tomcat 8.0.33 is the latest Tomcat 8 version.

3. Prior to starting the Web Server start the Administration Client and import the Thin Client properties located in the classes folder to the Central Configuration.
4. Restart the Tomcat server.
5. Once the SignPlus.war package has been deployed to the WebServer shutdown the Tomcat server and again start the Admin Client. Now the required properties for the Business model must be imported for the Thin Client or set to be used from a previous setup for the Business Model of the current Thin Client instance.

Important Thin Client properties are always set for instance type 'ThinClient'. Should a specific set of properties be required for a unique Thin Client server then this can be achieved by setting a unique instance name in the respective **web.xml** property 'InstanceName' as described in the [Servlet Parameter Thin Client](#) section.

The Tomcat server will automatically deploy all components as required and a new web context will be created under the:

```
%CATALINA_HOME%\webapps
```

called **SignPlus** (by default).

In older installations (pre release 3.6), it was necessary to configure Tomcat's **setclasspath.bat** so that shared components could be found. Since all required components are delivered as part of the web archive (.war file), this is no longer necessary. However, it is important that the binary files can be found on the system PATH. For this reason, the **setenv.bat** is copied into the bin directory of

the tomcat installation, where it is automatically called by **catalina.bat** (if available). Alternative the Windows **PATH** environment variable is adjusted with the following entry:

```
%CATALINA_HOME%\webapps\SignPlus\WEB-INF\lib\bin
```

In addition, useful JVM (Java Virtual Machine) start parameters are also set in **setenv.bat**.

```
SET JAVA_OPTS=-server -Xss64k -Xms50m -Xmx512m -Djava.library.path=%  
CATALINA_HOME%\webapps\SignPlus\WEB-INF\lib\bin
```

The **java.library.path** is a mandatory parameter. Otherwise it could happen that necessary DLLs (dynamic link libraries) cannot be found during runtime, especially if more than one Thin Client context is implemented in one Tomcat instance or if several Tomcat instances are installed on the same machine.

The batch command file **setenv.bat** is only called by the Tomcat during bootstrap tomcat is started also as command via **startup.bat**. In production usage tomcat is rather started as Windows service whereas **java.library.path** just like the other java parameters has to be set in the tomcat control panel.

The context name may be changed by first changing the name of the **SignPlus.war** file to something else before copying it into the webapps directory. For example, if you would like the web context to be called **MyApp** instead of **SignPlus**, rename the file **SignPlus.war** to **MyApp.war** before step 1 above.

If you have already followed the procedure above for deploying the .war file and you would still like to rename the context, you may simply rename the:

```
%CATALINA_HOME%\webapps\SignPlus
```

folder to another name. Tomcat will always use the folder name as the web context name.

Do not forget to adjust the PATH variable or the **setenv.bat** file as well in order to point to the binary files in the

```
/MyApp/WEB-INF/lib/bin
```

directory.

Important Don't forget to set the PATH variable (or in the **setenv.bat** file) to the corresponding directory and to set **java.library.path** to ensure that programs are able to run in:

```
/SignPlus/WEB-INF/lib/bin
```

directory.

Updating a Thin Client installation

In order to update the Thin Client installation, do the following procedures:

1. Remove the old **SignPlus.war** file (or other WAR file if it has been renamed).
2. Remove the context folder:

```
%CATALINA_HOME%\webapps\SignPlus
```

3. Copy the new SignPlus.war file into

```
%CATALINA_HOME%\webapps
```

4. Should any property changes be required the appropriate properties must be updated in the Central Configuration before restarting.
5. Restart the Tomcat server.

Starting Tomcat as Service

In order to start Tomcat as a service follow the implementation steps described in Tomcat "Windows service HOW-TO" documentation (<http://tomcat.apache.org>).

Web browser configuration

In order to use the Thin Client a browser will be required to display the data. The Internet Explorer and Mozilla (or Firefox) browsers are supported for FraudOne and require configuration depending on the version selected to be used. FraudOne doesn't offer any support for any further browsers. The major requirements for both browsers is that:

1. For the first installation or usage of the browser with the FraudOne Thin Client the browser must be started **Run as administrator**. This is required to allow the plug-in or ActiveX to be installed.
2. Plug-ins (plug-in for Firefox or ActiveX for Internet Explorer) must be allowed / supported. There is a further description of this for various browsers and browser versions below.
3. JavaScript must be enabled. For some browser versions this may need to be selected in the respective browser properties.
4. For the Internet Explorer you may also need to allow ActiveX (active content) to run on your computer.

Additional information for the various browsers and browser versions

Internet Explorer (referred to as IE) requires the Compatibility Mode to be set when the ActiveX is required. This is the case when a SignPad (tablet) is connected for verifying signatures or if the client side logon is desired. Otherwise the Compatibility Mode will be controlled by the server and no settings are required in the IE browser for this.

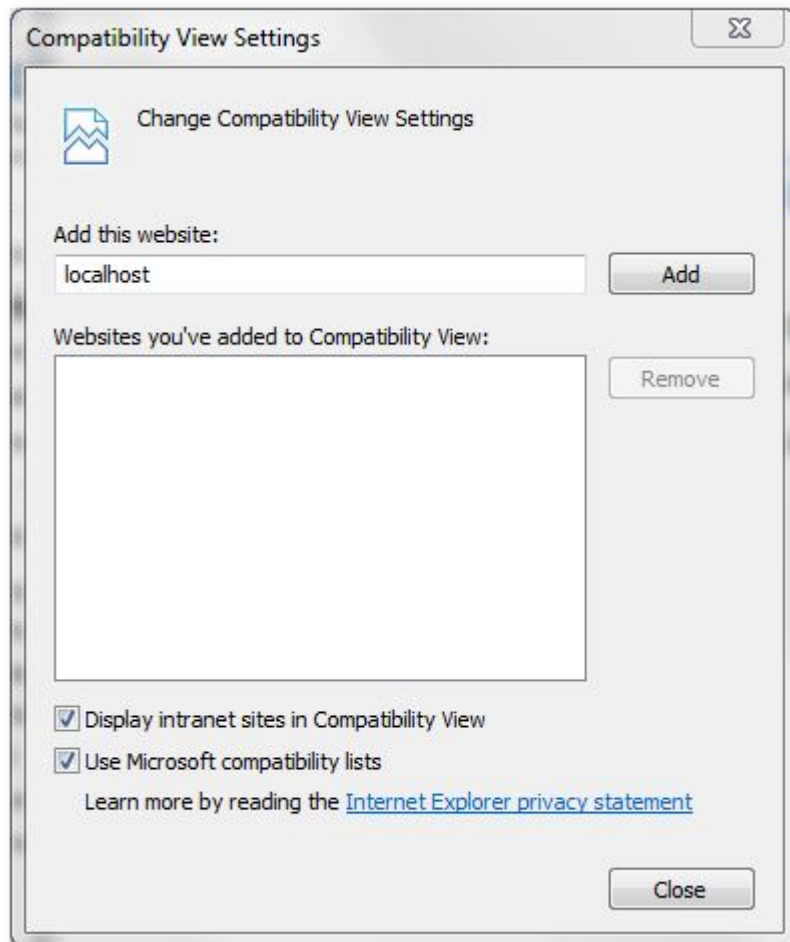
Should the ActiveX or Plugin be required continue with the appropriate steps described below.

IE 11.x, 10.x, 9.x

Perform the following steps if ActiveX is required:

1. Open the IE.
2. Select **Compatibility View Settings** from the **Tools** menu or icon.

The following dialog will appear:

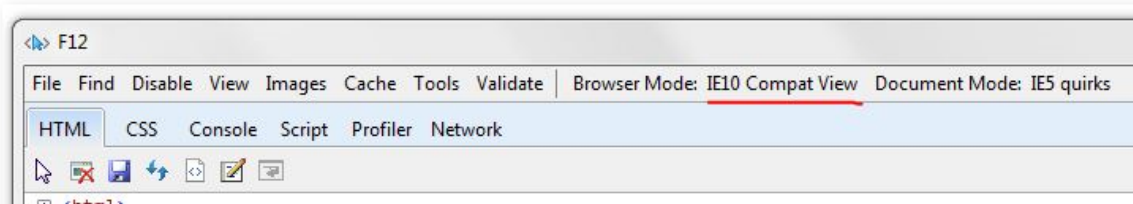


3. Input the name of the website to which the Compatibility View is required. This will be the server you are connected to where your Web Service is running for the FraudOne Thin Client (ie: localhost, myServer ...).
4. Select **Close** and the compatibility mode has been set.

IE 10.x (optional)

The IE 10 must be set to **Compatibility Mode** to function properly with the FraudOne Thin Client. This is easily performed by:

1. Open the IE 10.
2. Press **F12** or select **F12 Developer Tools** in the **Settings** icon or menu when the IE is running.
3. A window will open as shown below.
4. Select **IE10 Compat View** for the **Browser Mode**.



Firefox (38 –) 45 and Firefox ESR 45.x

When using Firefox there is a setting required for activating the plug-in loaded by the FraudOne Thin Client. This setting is required on Firefox versions 38 – 45. Following Version 45 the Firefox browser will be supported only as an ESR 45.x release.

[Exert from Mozilla Internet page:](#)

Override add-on signing (advanced users)

You can temporarily override the setting to enforce the add-on signing requirement by changing the preference `xpinstall.signatures.required` to `false` in the [Firefox Configuration Editor](#) (`about:config` page). Support is not available for any changes made with the Configuration Editor so please do this at your own risk. Signing will be mandatory with no override, in Firefox 46 beta and release versions. For details, see [this Mozilla blog](#).

Use the link in the exert above for further information on setting the `xpinstall.signatures.required` to 'false' property in the 'about:config' properties for Firefox.

Note Firefox 46.x and higher are no longer supported by FraudOne!

Manual registration of Thin Client ActiveX Controls

The following functionalities provided by the Thin Client require the deployment of Active-X controls on the client machine:

1. Pad Verification (i.e. Sign Teller Verification)
2. C-API compatibility interface for Teller mode
3. Client-side logon
4. Usage of Instant Check Verification (ICV)

The default behavior is that the ActiveX components will be downloaded on-demand when the client accesses a web page that requires the component. However, in some installations it may be desirable to perform this step manually (e.g. if the browser is setup to disable ActiveX downloading).

If a manual installation of the required ActiveX controls/plug-ins for the Thin Client is required, the following command should be executed in a Windows DOS box:

```
regsvr32.exe activeTC.dll
```

This assumes that the **activeTC.dll** (the plug-in) resides in the current folder where the command is executed. Note that in order to execute this command, the user requires special rights.

To enable automatic downloading of plug-ins, make sure the following Internet options have been set for Internet Explorer.

ActiveX controls and plug-ins:

'Download **signed** ActiveX controls' = **Enable**

'Download **unsigned** ActiveX controls' = **Disable**

'Script ActiveX controls marked as safe for scripting' = **Enable**

'Initialize and script ActiveX controls not marked as safe' = **Disable**

'Run ActiveX controls and plug-ins' = **Enable**

URI encoding for German Umlauts

For a German name search within the Thin Client you must enter sometimes also umlauts in the search dialog.

Since Tomcat V5.x it is necessary to enter the

```
URIEncoding = "UTF-8"
```

attribute within the <Connector> tags in the

```
%CATALINA_HOME%\conf\server.xml
```

configuration file for correct handling of these umlauts at the Business Model Server.

Using the Thin Client

Operation Modes

The client may be configured to operate in standard Thin Client or in Teller mode.

Standard Thin Client

This mode is the default mode and is a basic web application with direct user interaction within a browser.

The start page of the Thin Client application in the standard operation mode may be accessed using any of the following URLs:

1. The following URL is used for backward compatibility with pre 3.6 releases:

```
http://<yourhost>/SignPlus/StartUp.jsp
```

2. Recommended URL:

```
http://<yourhost>/SignPlus/logon.do
```

3. The following URL starts the welcome page

```
http://<yourhost>/SignPlus[/index.jsp]
```

You need an additional click in order to get the login page:

Teller Mode

This mode can be configured to allow external applications to access FraudOne functionality directly without user interaction. There are two external APIs available. The first is the C-API, also known as compatibility interface, is a programmatic interface. The second API is a servlet-based interface. The servlet interface can be used within another web application by calling URLs that adhere to the servlet interface specification.

The **tcustom.properties** file detailed in [Configuring the Thin Client](#) section contains the **IsTeller** settings key which enables the Teller mode if it is set to true.

Important Since Teller mode and default mode cannot run in the same context, it is necessary to install the FraudOne Thin Client application (Business Model Server) twice. Two types of installation are possible to run the different modes parallel.

Each operation mode must reside in its own servlet container instance on the web server. I.e. you have to install one Tomcat instance for each of the Thin Client modes.

If you want to run the two FraudOne Thin Clients within one Tomcat instance, each of them needs its own context. You can reach this by copying the **SignPlus.war** file with two different names into the **webapps** directory of the Tomcat (e.g. **SignPlus.war** and **SignPlus1.war**). In order to run one FraudOne instance in Teller mode, you have to change the **IsTeller** entry in **tcustom.properties** to (see section Teller Mode):

```
IsTeller = true
```

Furthermore you have to copy **spnative.jar** from:

```
%CATALINA_HOME%\webapps\SignPlus\WEB-INF\lib
```

directory into the:

```
%CATALINA_HOME%\shared\lib
```

directory.

The spnative.jar in both

```
%CATALINA_HOME%\webapps\SignPlusX\WEB-INF\lib
```

directories has to be deleted then!

In Teller mode the Thin Client may be accessed using the following URLs:

1. The following URL is for access via the Teller's servlet interface. The parameters available on the URL must adhere to the API specification.

```
http://<yourhost>/SignPlus/SPTellerEntry?yourparameter=yourvalue
```

An example URL to search for Account 11111 would then look like:

```
http://<yourhost>/SignPlus/SPTellerEntry?
  AccountNo=11111&CustomerNo=11111&BNO=001&CountryID=049&BankCode=70
  6&ReturnTo=http://someurlToReturnTo&Fct=01
```

2. The following URL is for access via the compatibility interface (C-API).

```
http://<yourhost>/SignPlus/StartTeller.jsp
```

Further information about the usage of the Teller interfaces can be found in the document *Standard Teller Interface*.

Application Modes

The Thin Client may present different application modes to the user. These modes provide a different view of the same data and depending on the mode, restricted functionality is available. The application modes are the following:

- Standard Thin Client

- SignInfo Classic Edition (SICE) Client

Standard Thin Client

The standard Thin Client provides the following FraudOne functionalities available via the web application including:

SignBase: Read only access

SignInfo: Read only access

SignCheck: all FraudOne functionalities

Exactly what is available to the user is configured via customizable properties, and determined by license availability.

SignInfo Classic Edition (SICE) Client

The SICE Client is a stripped-down version of the standard Thin Client providing the loading of images for display from the SignInfo database. It is not the same as the standard Thin Client SignInfo "view" and is used only if a very basic view onto customer/account images is desired.

Both of these modes may be used in mixed combinations with the operation modes (Teller and standard Thin Client). The **tcustom.properties** file detailed in 'Configuring the Thin Client' section below contains the **SICE.Enabled** which initiates the Thin Client in SICE mode.

Configuring the Thin Client

Once the Thin Client has been installed on a server, its configuration may be altered by editing the property elements that are found in the central configuration server and can be managed using the Administration Client:

1. **tcustom.properties**: the main customization element file for the Thin Client. Contains all properties specific to the Thin Client such as enabled functionality, layout etc.
2. **custom.properties**: FraudOne Business Model properties. Contains all server and Business Model specific properties such as host address of the application server, the definition of used components (SignBase / SignCheck / SignInfo) or the logon options. For more information see the Java Client **custom.properties** description in [The custom.properties configuration file](#).
3. **visibility.properties**: Specifies which default FraudOne fields are visible within the Thin Client application.

tcustom.properties

The **tcustom.properties** configuration element allows setting the parameters for the configuration of the Thin Client.

The following describes all of the basic properties.

General Thin Client settings

The following parameters can be used for general Thin Client settings:

[SICE.Enabled](#) = false

```
BankApplicationName = "SignPlus DemoBank"
BankSiceApplicationName = "SignInfo Classic Edition for Demo Bank"
SignatoryNameClipLength = 30
HandleSessionTimeout = true
DisableBrowserShortcuts = false
BrowserShortcut =
DisableRightMouseClick = false
DisableBrowserToolbars = false
NewWindowWidth = 800
NewWindowHeight = 600
DisableToolTips = false
DisableDrag = false
SignInfoShowDetailsOnRight = true
SignInfoCleanImagePreview = false
CheckStockShowDetailsOnRight = true
ImageSmoothing = 1
jpegQuality = 0.04
ChangePwd = 1
displayLogo = 1
LogoImageName = /res/logo2.gif
help.de.thin = help/de/thin/default.htm
help.en.thin = help/en/thin/default.htm
help.de.sice = help/de/sice/default.htm
help.en.sice = help/en/sice/default.htm
EnablePhysicalVerification = false
UseSTV = 0
UseVpsv = false
UseICV = false
TwainLeft = 0
TwainRight = 0
TwainTop = 0
TwainBottom = 7.3
TwainUnits = 0
TwainResolution = 100.0
SetTwainAutoBright = true
SetTwainAutoDeskew = true
SetTwainThreshold = 128
SetTwainProgress = true
ScaleToGray = true
DisableAllMenuEntries = false
EnableFontSelect = true
EnableLogoffMenu = true
EnableAboutMenu = true
DisableHelpButton = false
UseActiveX = 0
LogFile =
LogLevel = severe
useLeadTools = 0
```

Thin Client settings for SignBase display

The following parameters can be used for Thin Client settings concerning SignBase Display:

```
DefaultShowStockImage = 1
DefaultShowSignInfo = 1
DefaultShowSignatory = 1
SignatoryNameSignatoryListClipLength = 0
RulesTextSignatoryListClipLength = 90
DisableImageTools = false
EnableThinClientShortcuts =
```

```
EnableSignatureCapture = false
EnableSignatureReplace = false
PadBackImage1 = PadBack1.bmp
PadBackImage2 = PadBack2.bmp
PadBackImage3 = PadBack3.bmp
PadBackImage1.widthxheight = PadBack1.bmp
PadBackImage2.widthxheight = PadBack2.bmp
PadBackImage3.widthxheight = PadBack3.bmp
```

Thin Client settings for SignCheck

The following parameters can be used for Thin Client settings concerning SignCheck:

```
SignCheckDisableImageTools = false
EnableSignCheckHistory = false
EnableFullDocumentHistory = true
SignCheckShowHistoryData = true
EnableSignCheckStatus = true
EnableSignCheckLists = true
EnableSignCheckQueues = true
EnableStoreAsVariant = true
EnableSignCheckVBD = true
VBDQueueName = VBD
EnableQueueRightsForStatus = true
EnablePrimanota = true
SignCheckShowCheckData = false
EnableSignCheckViewPersistence =
SignCheckNotify = false
SignCheckNotifyApplet = true
SignCheckNotifyInterval = 30
EnableThinClientShortcuts =
GetSpecificLock = true
```

Teller specific Thin Client settings

The following parameters can be used for Teller-specific Thin Client settings:

```
IsTeller = false
EnableTellerErrorDialog = false
CloseTellerPopupOpener = true
Kld2TellerStyle = /jsp/styles/SignPlus-styles-xs.css
Ups2TellerStyle = /jsp/styles/SignPlus-styles-s.css
SPTellerEntryStyle = /jsp/styles/SignPlus-styles-s.css
SignatoryNameClipLength.Teller =
SignatoryNameClipLength.Teller.Kld2Teller =
SignatoryNameClipLength.Teller.Ups2Teller =
SignatoryNameSignatoryListClipLength.Teller =
SignatoryNameSignatoryListClipLength.Teller.Kld2Teller =
SignatoryNameSignatoryListClipLength.Teller.Ups2Teller =
RulesTextSignatoryListClipLength.Teller =
RulesTextSignatoryListClipLength.Teller.Kld2Teller =
RulesTextSignatoryListClipLength.Teller.Ups2Teller =
EnableTellerShortcuts =
displayLogo.Teller =
displayLogo.Kld2Teller =
displayLogo.Ups2Teller =
```

```
DisableAllMenuEntries.Teller =  
DisableAllMenuEntries.Teller.Kld2Teller =  
DisableAllMenuEntries.Teller.Ups2Teller =  
Teller.Request.Image.Transfer.Format = base64  
EnableAboutMenu.Teller =  
EnableAboutMenu.Teller.Kld2Teller =  
EnableAboutMenu.Teller.Ups2Teller =  
DisableHelpButton.Teller =  
DisableHelpButton.Teller.Kld2Teller =  
DisableHelpButton.Teller.Ups2Teller =  
Teller.Search.RemoveZeros.AccountNo = false  
Teller.Search.RemoveZeros.CustomerNo = false
```

LDAP specific Thin Client settings

Setting up an LDAP server login enables an LDAP server to verify the username and password. A login entry dialog box is displayed to the user but the authentication is done via the LDAP server before connecting to the SignBase server.

```
UseLDAP =  
LDAP.Version =  
LDAP.Host =  
LDAP.Port  
LDAP.Context  
LDAP.DN = host=tsys,o=yourbank,c=us,cn=localhost  
LDAP.Objectclass =
```

Data Warehouse specific Thin Client settings

Setting up a FraudOne Data Warehouse enables to archive check data, which was processed in FraudOne. Some settings have to be made in **tcustom.properties** file to get access to this short term archive. More information can be found in the "Release 4.0 – Feature Description".

```
SCArchive.Connect.DBMS =  
SCArchive.Connect.DBName =  
SCArchive.Connect.DBServer =  
SCArchive.Connect.Schema =  
SCArchive.Connect.User =  
SCArchive.Connect.PW =  
SCArchive.Image.Enabled =  
SCArchive.Image.Impl =  
SCArchive.Resultlist.MaxRows =
```

JDBC specific settings for short term archive access:

```
JDBCClass.DB2UDB =  
JDBCClass.DB2ZOS =  
JDBCClass.ORACLE-OCI =  
JDBCClass.ORACLE-THIN =  
JDBCClass.MSSQL =  
JDBCURL.DB2UDB =  
JDBCURL.DB2ZOS =  
JDBCURL.ORACLE-OCI =  
JDBCURL.ORACLE-THIN =  
JDBCURL.MSSQL =
```


Important

%(d) will be replaced during runtime with the value from entry **SCArchive.Connect.DBName**.
 %(s) will be replaced during runtime with the value from entry **SCArchive.Connect.DBServer** (TCP/IP hostname).

Image Zooming Dimensions and Scaling Strategies

The MaxWidth property specifies the maximum width in the given context.

The MaxHeight property specifies the maximum height in the given context.

The ScalingStrategy property specifies the strategy to be used for scaling in the given context. Possible values are:

- Scale to width and height of browser window (default):

```
xxx.ScalingStrategy = 0
```

- Scale to width of browser window

```
xxx.ScalingStrategy = 1
```

- Scale to height of browser window

```
xxx.ScalingStrategy = 2
```

Strategy 0 should be used for applications where images are mixed (both portrait and landscape).

Strategy 1 should be used for images that are mostly portrait (i.e. width is greater than height).

Strategy 2 should be used for images that are mostly landscape (i.e. height is greater than width).

```
SignatoryListView.MaxWidth = 200
SignatoryListView.MaxHeight = 100
SignatoryListView.ScalingStrategy = 0
```

```
ComplexRulesView.MaxWidth = 300
ComplexRulesView.MaxHeight = 300
ComplexRulesView.ScalingStrategy = 0
```

```
SignatoryDetailsView.MaxWidth = 350
SignatoryDetailsView.MaxHeight = 100
SignatoryDetailsView.ScalingStrategy = 0
```

```
SignWithShortList.MaxWidth = 80
SignWithShortList.MaxHeight = 80
SignWithShortList.ScalingStrategy = 0
```

```
ImageView.MaxWidth = 500
ImageView.MaxHeight = 400
ImageView.ScalingStrategy = 0
```

```
SignInfoImagePreview.MaxWidth = 150
SignInfoImagePreview.MaxHeight = 150
SignInfoImagePreview.ScalingStrategy = 0
```

```
VpsvView.MaxWidth = 200
VpsvView.MaxHeight = 100
VpsvView.ScalingStrategy = 0
```

```
SignCheckDocumentFront.MaxWidth = 400
SignCheckDocumentFront.MaxHeight = 300
SignCheckDocumentFront.ScalingStrategy = 0
```

```

SignCheckDocumentBack.MaxWidth = 400
SignCheckDocumentBack.MaxHeight = 300
SignCheckDocumentBack.ScalingStrategy = 0

SignCheckDocumentSignature.MaxWidth = 400
SignCheckDocumentSignature.MaxHeight = 200
SignCheckDocumentSignature.ScalingStrategy = 0

StoreVariantView.MaxWidth = 800
StoreVariantView.MaxHeight = 300

SiceResultList.MaxHeight = 200
SiceResultList.MaxWidth = 200
SiceResultList.ScalingStrategy = 0

CheckStockResultList.MaxHeight = 200
CheckStockResultList.MaxWidth = 400
CheckStockResultList.ScalingStrategy = 1

CheckStockResultListPreview.MaxHeight = 300
CheckStockResultListPreview.MaxWidth = 700
CheckStockResultListPreview.ScalingStrategy = 1

```

Plugin Properties

In order to prevent an auto-update of the Plugin/ActiveX component (see also [Manual registration of Thin Client ActiveX controls](#)), the following properties have been added to the Thin Client configuration file **tcustom.properties**:

```

ActiveXVersion = 2,1,8,0
PluginVersion.Major = 2
PluginVersion.Minor = 1
PluginVersion.Revision = 8
PluginVersion.Build = 0

```

If the client has no ActiveX/Plugin yet installed, the version available on the Web Server (Business Model Server) will be installed. If the client has a newer ActiveX/Plugin already installed, no update is done.

If the client has an older ActiveX/Plugin version installed than configured via the properties, the client is updated with the newer version from Web Server (Business Model Server) - if the Web Server can provide a newer version.

Web Server Side Login

The Thin Client may be configured to perform a "server side" login (see subchapter **custom.properties**) and set the rights for each BNO.

If server side login is requested (UseSignSec=0 in **custom.properties**), the following parameters are read from **tcustom.properties** in order to perform a FraudOne login. In order to accomplish a valid logon the user must be configured in the FraudOne user administration.

Parameter	Description
Login.User Login.Password	Standard user and password, with whom FraudOne login is performed

Parameter	Description
	(no default user and password)

There are two possibilities to get the rights of this user.

1. The rights of this user are configured also in FraudOne user administration and will be read after logon from the application server. Then if

```
VerifyLogData=1
```

must be set in the **custom.properties** file.

2. The user rights are also set in **tcustom.properties** file (to obtain these rights you have to set

```
VerifyLogData = 0
```

in the **custom.properties** file. The rights can be set for each bank number, which is set in the list for key BNO.List (**custom.properties**). The rights must be set in pairs for each BNO.

Parameter	Description
SBRights.xxx	xxx as the second part of the key defines the BNO, for which the SignBase/SignCheck right should be set. BNO 999 means, that the defined rights should be set for all BNO's. The value for these keys is a number, which describes the kind of rights.
SCRights.xxx	<p>Samples</p> <pre>SBRights.001 = 129023 SCRights.001 = 129023 SBRights.002 = 0 SCRights.002 = 0</pre> <p>Default: no rights defined</p>

The numbers representing the rights are to be calculated and set as follows:

```
# constants for user rights SignBase, bitwise set
SB_INQUIRY      = 0x00000001      1
SB_CHANGE      = 0x00000002
SB_VERIFY      = 0x00000004
SB_ADMIN      = 0x00000008
SB_Audit      = 0x00000010
SB_STV      = 0x00000020      32
SB_Report      = 0x00000040
-----
--
33 -> SBRights.xxx=33

# constants for user rights SignCheck, bitwise set
SC_GROUP1      = 0x00000001
SC_GROUP2      = 0x00000002      2
SC_PTSV      = 0x00000004
SC_VSM      = 0x00000008
SC_VC      = 0x00000010
SC_VARIANT      = 0x00000020      32
SC_STATS      = 0x00000040
```

```

SD_VBD          = 0x00000080
SD_PRINT        = 0x00000100
SD_OPERATOR     = 0x00000200
SC_ADMIN        = 0x00000400
SC_INQUIRY      = 0x00000800          2048
SC_GROUP3       = 0x00001000
SC_GROUP4       = 0x00002000
SC_PTSV2        = 0x00004000
SC_CHANGEDOC    = 0x00008000          32768
SC_VIEWDECISIONS = 0x00010000;

-----
--
                                     34850 -> SCRights.xxx=34850

```

If the rights of the server side login user are read from **tcustom.properties**, you can set more default values for this user, which overrides the settings from **custom.properties**.

[Login.Bno](#)
[Login.Cid](#)
[Login.BankCode](#)
[Login.BranchCode](#)

Thin Client Specific Business Model settings in custom.properties

The Thin Client requires components of the FraudOne Business Model also used by the Java Client. Some entries of the file **custom.properties** are not only used by the Business Model but also additionally by the Thin Client directly. This configuration element must reside in the central configuration server.

The **custom.properties** file is the main configuration file for the FraudOne Business Model including host configuration and login type. Three login types are supported:

Parameter	Description
UseSignSec =	0 - Server side login (users are logged in with server given predefined rights) 1 - Default login with user name and password 2 - Client side login (login data provided by browser plugin on the client-side) Default: 1
VerifyLogData =	Value is interpreted for client or server side login: 0 - Rights are obtained from tcustom.properties (Business Model Server). No FraudOne login is performed. 1 - Rights are obtained from FraudOne user administration after login. 2 - Rights are obtained from client side dll (browser) for client side login. No FraudOne login is performed. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> Note VerifyLogData setting has no meaning for UseSignSec = 1 </div>

Standard Login

Standard Login with manual entry of userid and password in the login dialog is requested if

```
UseSignSec = 1
```

is set in **custom.properties** file. The user is verified against the FraudOne user administration. The user rights are received from the FraudOne application server after successful authentication.

Client-side Login

It is also possible to configure a client side login with

```
UseSignSec = 2
```

in the **custom.properties** file. Client side login means, that the authentication of the user is done on the client (browser) not on the server. So it is possible to integrate FraudOne in the bank's own bank wide security system. Since the action radius of the browser is very reduced, because of security reasons, it is necessary to realize a client side login with the help of a plugin or an ActiveX control for MS Internet Explorer.

The browser plugin loads a custom specific DLL, which must comply with a specified interface, which is described in detail in **sp_login_en.pdf** (User Login: Integration between Application Environment and SignBase). After successful authentication in this DLL the user information is send directly from the plugin to the FraudOne Business Model Server.

The next step is to get the rights for this user.

One possibility to acquire the rights is another call to the plugin on the client (browser) side, whereas the customer specific calls must be implemented also in the already mentioned DLL.

The user rights are submitted subsequently again directly from the plugin to the Web Server. Prerequisite is

```
VerifyLogData = 2
```

in the **custom.properties** file.

No (implicit) login to SignBase is required.

Another way to get user rights after a client side login is the inquiry (implicit login) to SignBase Application Server for the identified user. It implies that the rights of all users have to be maintained in FraudOne. This can be achieved with

```
VerifyLogData = 1
```

setting in **custom.properties**.

Supplement:

It is possible to define a common password for all (client side identified) users, which is used for each SignBase login. This feature is activated with setting

```
ServerSideUserAuthenticator = 1
```

in the **tcustom.properties** file. In this case the common password can be configured with `Login.Password=...` setting in **tcustom.properties**.

With

```
VerifyLogData = 0
```

in **custom.properties** the SB- and SC-rights are obtained from **tcustom.properties** without additional (implicit) login to SignBase. The rights can be differenced there by BNO (Bank Number) but not by user. For more information see chapter [Web Server side login](#).

Check account number

Equivalent to the Java Client it is possible to verify the passed customer or account number against a customizable checksum algorithm. Since the Business Model is responsible for this check, it is the same parameter setting in the **custom.properties**, which activates this feature.

Parameter	Description
UseChecksum =	1 - Checksum calculation for customer and/or account (bank-specific configurable) 0 - No checksum calculation Default: 0

Logging

The Thin Client may be configured to perform logging of its internal functions for troubleshooting purposes. The logging may be activated either before startup statically, and/or dynamically during the operation of the system.

Static Logging Setup

This approach should be used if the logging is always to take place regardless of whether the Tomcat server has been shutdown and restarted. The output logfile is specified using the **LogFile** property in the **tcustom.properties** configuration file (see [General Thin Client settings](#)). The desired logging level is specified by setting the **LogLevel** property (also in **tcustom.properties**). The following settings are valid for **LogLevel**:

Log Level	Description
off	No logging is activated Default: off
fine	Debugging output only
info	Only informational output
warning	Only warnings will be output
severe	Only critical errors will be output
all	All logs will be activated

When logging information is output in the log file, it is prefixed with a single character denoting the type of log according to the following table:

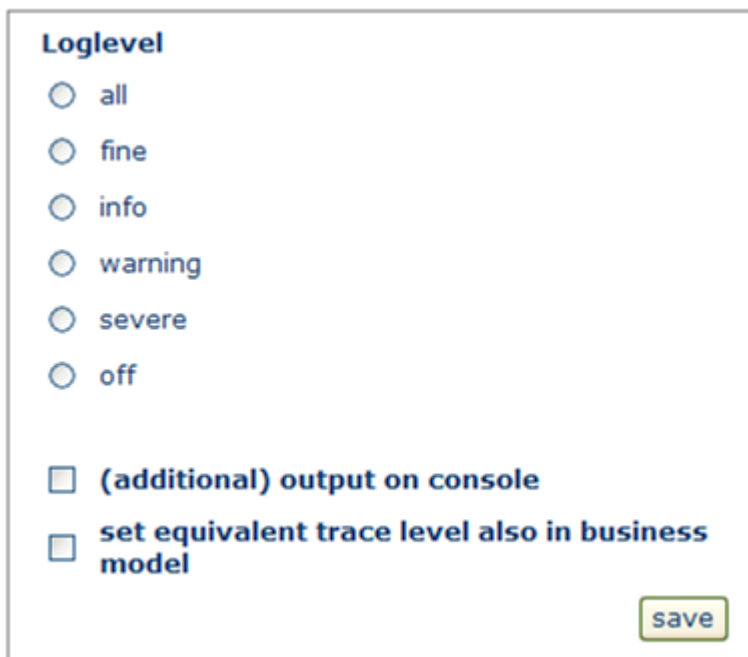
Log Level	Output prefix
off	N/A Default: off
fine	D
info	I
warning	E
severe	C
all	Dependent on log output type (see above levels)

Dynamic Logging Setup

In addition to the static method, logging may be activated dynamically on an already running system. To do this, request the following URL via any web browser:

Http://<yourhost>:<port>/<YourSignPlusContext>/logger.jsp

where <YourSignPlusContext> is the root context of your FraudOne Thin Client installation. The **logger.jsp** will return a page which looks as follows:



Loglevel

☐ all

☐ fine

☐ info

☐ warning

☐ severe

☐ off

☐ **(additional) output on console**

☐ **set equivalent trace level also in business model**

save

The Log level is then set by activating the appropriate radio button. The levels are as described in the previous section. If the **(additional) output on console** checkbox is checked, the logging output will also be sent to the Tomcat console. The settings will be immediately in effect after clicking the **save** button.

Using the dynamic method, only the logging level may be modified. The output log file to be used may only be modified via the LogFile property prior to system startup.

It is also possible to change logging options dynamically for the underlying Business Model by clicking on the checkbox **set equivalent trace level also in business model**.

Log Level	Trace level for Business Model
off (default)	0
fine	4
info	8
warning	2
severe	1
all	-1

Fix logging values for the Business Model, which are valid right from the start of Tomcat, can be set in the **custom.properties** element located in the central configuration server.

See chapter Configuring the Java Client, [The custom.properties configuration file](#).

Important If the Tomcat server is restarted, the log settings will revert to the default values specified in the **tcustom.properties** and custom.properties file. I.e. configuration settings which have been modified via the **logger.jsp** are valid only for the current server execution and will not be persisted.

Limitations of the Thin Client

The number of Thin Client instances as well as the number of concurrent sessions determines the amount of Tomcat instances needed. This may be dependent on the following factors:

- The amount of concurrent requests (frequency) coming from the Thin Client within one standard session
- Whether the Thin Client is used for SignBase query or for SignCheck verification
- Only one instance of Java Client can be operated per workstation

Contact your Kofax business representative for a more accurate Thin Client representation.

SignCheck Engines

General

This chapter describes the SignCheck Engines in the standard functionality as well as the configurations that can be performed by an administrator. The availability of certain Engines in your installation may vary depending on the products you have purchased.

Overview SignCheck Engines

The SignCheck automatic verification process is used to increase the efficiency and security of the whole verification workflow. The SignCheck Engines provide a combination of automatic verification processes that can be implemented into the workflow.

The SignCheck main module with its built-in SignCheck Workflow provides a central framework for complex item review operations. Check items (data and images) are loaded into the SignCheck database for processing and are processed as part of a verification workflow. The exact flow of the items during the verification process will vary depending on the customer's specific needs.

Depending on the required throughput (the amount of items processed in a predefined time), additional verification processes can be added. Within the process of automatic verification, a match rate is determined and compared to a predefined acceptance rate. Different levels of required match rates can be configured depending on the amount found in a transaction. Usually, an item rejected by an automatic verification process is moved to a visual verification queue to be verified by a human user.

The types of available SignCheck Engines are as follows:

- **ASV:** Automatic Signature Verification.
Compares the signature snippet from the check against the signature images in the SignBase database.
- **GIA:** Generic Image Analysis.
Compares the check image against reference check stock images in the SignBase database.
- **ARV:** Automatic Rule Verification.
Compare rules of user selected signatories against the rules of the check account.
- **APSV:** Automatic Payment Signature Verification.
Provides live verification of signatures provided on a front office signature pad.

Centralized configuration

The configuration information for the SignCheck Engines can be stored in a centrally managed and audited database, the engines themselves then read their configuration information from the database rather than from the file **automat2.ini** that is described in this document.

The FraudOne Administration Client is used to administer the contents of the database used for engine configuration. The access to the configuration data using the Administration Client is subject to user authentication and access permissions. Refer to the documentation for the [FraudOne Administration Client](#) for the description of the administration process.

Configuration information

All configuration items and sections described in this document as being part of the **automat2.ini** configuration file are stored in the centralized configuration database tables if this feature is enabled. The items and sections themselves are unchanged from their description in this document.

Prerequisite configuration actions

Before the centralized configuration feature can be used, it is necessary to carry out an installation process on each machine where an Automat (ASV, GIA etc.) Server is installed. It is not necessary to carry out the procedure on machines where only SignCheck Engine processes are installed. See the chapter on Server Manager Administration for the required installation process for the SignCheck Automat Server machines.

SignCheck Engines

The SignCheck Engines that are available for a customer installation can vary. This section describes the different types of Engines that are available.

Automatic verification can be initiated on computers that are connected to the same network (LAN) as the Engine servers. The SignCheck Engines are located in the SignPlus directory or can be stored in a separate accessible directory. The SignCheck Engines carry out verifications as soon as there are document items in the SignCheck database with the status "ready" for the particular queue. The Engines request "ready" document items at preset intervals which can be set in the **automat2.ini** configuration file. The SignCheck Workflow Router locates the next available item for processing.

The diagram below depicts the processing workflow for the SignCheck Engines (except APSV Engine):

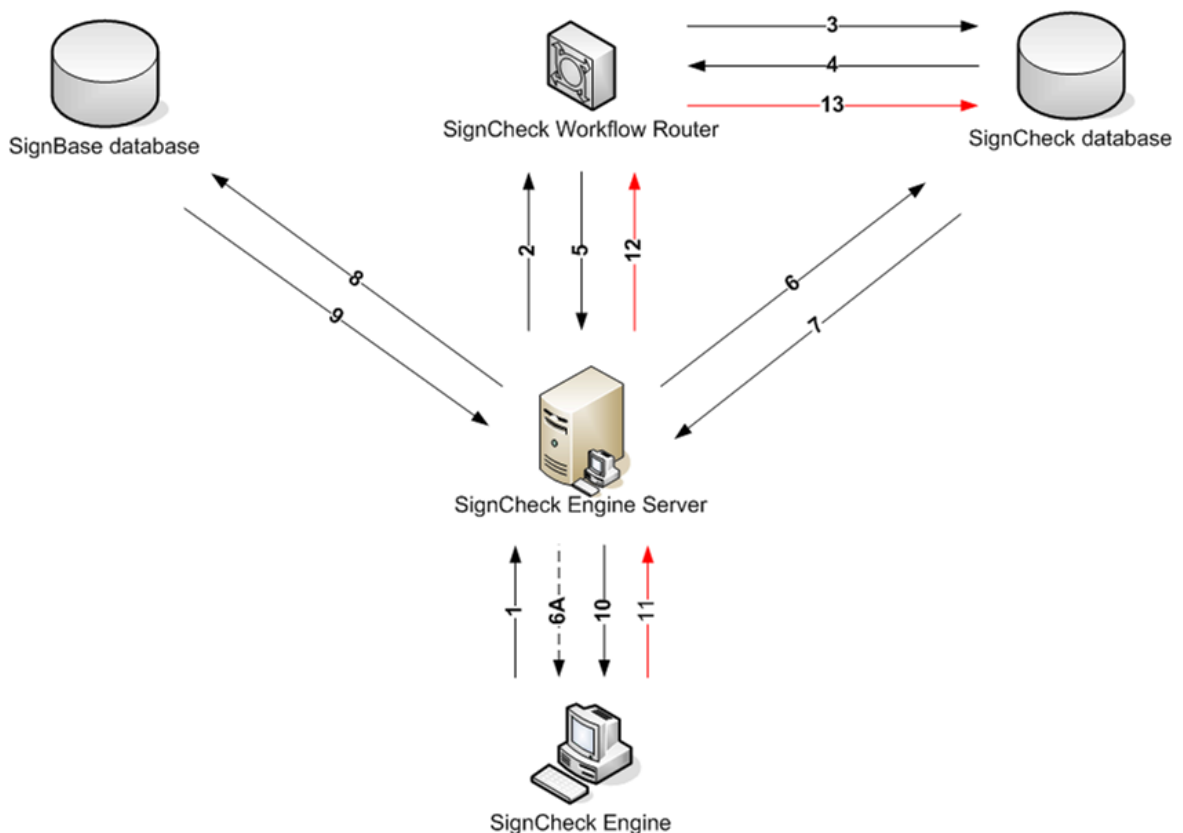


Figure 15: SignCheck Engine processing workflow

Where:

Number	Description
1	SignCheck Engine requests the next item for processing.
2	SignCheck Engine Server requests the next document item available for processing in its particular queue.
3	The SignCheck Workflow searches in the SignCheck database for the next item that is "ready" for that queue.
4	The SignCheck database sends a unique document identifier or "no item" back to the Workflow.
5	Workflow sends the unique document identifier or "no item" back to the Engine server.
6A	If there is nothing ready for processing then "no item" message is sent back to the Engine. The next request is sent after the predefined interval.
6	The SignCheck Engine server requests the information for the particular document identifier from the SignCheck database.
7	The information is sent to the Engine server.
8	The Engine then request the reference data for the particular account from the SignBase database.
9	The information is sent back to the Engine server.
10	The SignCheck and SignBase data is then sent to the SignCheck Engine for verification.
11	The SignCheck Engine sets a result of the item and sends it to the SignCheck server which is then set in the SignCheck database (in the SignCheck Workflow table).
12	The Engine server passes the result to the Workflow.
13	The Workflow updates the SignCheck database and performs the result-dependant routing.

Automatic Signature Verification (ASV)

Automatic Signature Verification provides an automated process that is used for the verification of signatures against reference data.

The ASV Engine compares signatures cropped from check images against a set of reference signatures in the SignBase database. The SignCheck ASV server receives the signature cropping of the check from the SignCheck database along with the customer identifier (customer number). The customer identifier is used to retrieve the corresponding customer information (customer account info, amount, signatures, signing rules, and currency) from the SignBase database. The ASV server only sends the relevant customer data along with the signature snippet and the check data to the SignCheck Engine for processing.

The SignCheck Engine process searches for a matching signature from the customer data received as well as the related signing rules. A result is based on the combination of "match rate" and rules verification for that particular signatory. If a signature match is found then a rules verification is performed (amount limits, time limits, group signing rules, and expiration times).

A match rate is determined for each comparison of the check signature image with each available reference signature. If no match rate is achieved then the ASV Engine compares the split parts of the check signature image with all available reference signatures.

The most important factors for a good match rate is the quality of the reference signatures in SignBase and a well cleaned and well positioned signature snippet from the payment forms.

Match Rates ASV

Each comparison of a check signature image with a reference signature delivers a separate result. This result is then verified against the configured threshold for that particular check amount (amount limits). There are ten different amount thresholds that can be initially configured by Kofax which range in carrying degrees from AA (signature image identified on the document shows a high similarity to that on the stored reference signature) to F4 (very little similarity could be found between the compared images).

Example

Having the key settings (currency: Dollar):

```
Limit-1 = 200000:C1
```

would mean that for up to 200,000 cents (2,000 Dollars) a minimum match rate of at least a C1 is required. Depending on the pre-configured acceptance rate (match rate with amount limit) determines whether to pass the item to the rules verification.

These thresholds configured in the **automat2.ini** configuration file under the appropriate SignCheck Engine and the following rules verification determines if a check is accepted or rejected and then routed to a Visual Verification process.

SIVAL Neural Networks Engine

The SignCheck ASV Engine uses the SIVAL neural network verification engine.

The SIVAL engine extracts the specific characteristics from a signature which is distinctive for that particular person's signature.

The SIVAL engine uses these unique characteristics for comparison and returns a result in a range of probability.

The SIVAL engine is initially customized by Kofax.

SIVAL split looks for a gap and splits the cropping image. If there is no match with the complete image, it tries to find the relevant signature by splitting the crop area. This is initially configured by Kofax in the **automat2.ini**, consult technical representative before making any changes.

Example

Maximum number of vertical SIVAL splits:

```
NumFsxSplitsBeside = 4
```

Use Only Clean Signature Snippets

The Engine can be configured to use only clean snippets for comparison. In a few cases it might happen that parts of company stamps overlapping with the signature are recognized to be significant parts of the signature itself, resulting in false accepts.

To avoid this behavior, set

```
UseOnlyCleanedSnippets = Yes
```

so that the automat is forced to clean all snippets before comparing against master signatures.

Twin Detection

A Twin Detection is the situation that occurs when a given signature, that should be verified, matches with more than one reference signature it was tested against. Detecting Twins in the FraudOne system is an important feature to assign the correct signatory (a signature AND a set of signing rules) and to avoid false accepts. As all reference signatures are tested, the given signature is not allowed to match with more than one of them (with the exception of variants of a signatory), because a unique rule-assignment won't be possible then.

If such a Twin is detected, the check is rejected with the return code `SCA_R_TWIN`. This return code is also set when the signing rules require two signatures on a check and they match against each other with a better match rate than `TwinTestRating`.

Important In order to avoid twin detection, it is essential that absolutely no duplicate signatures are captured or stored per Account/Customer. If you need more than one signature per signatory, add a variant to this signatory.

To turn the `TwinTest` off (which targets only the Best Match), the `TwinTestRating` has to be set to F5 or AAA in the **automat2.ini** configuration file. For a check to be considered for Twin test, the signature(s) has to have a match rate that is higher than the `TwinTestRating`.

Generic Image Analysis (GIA)

The SignCheck GIA can handle different engines (currently PPE and SAE) that support the APIA interface.

The SignCheck GIA with the **PPE** engine can search and crop snippets from a check stock image and store these snippets for further processing in the database.

The SignCheck GIA with the **SAE** engine can carry out verifications based on historical and statistical data from the past.

The SignCheck GIA server receives the check image from the SignCheck database along with the customer identifier (customer number). The customer identifier is used to retrieve the corresponding customer information (customer account info, amount, check stock image) from the SignBase database. The GIA server sends only the relevant customer data along with the check stock image to the SignCheck GIA Engine for processing.

The GIA Engine searches extended information on the check. Refer to [SignCheck Engines](#) for more information on the SignCheck Engine processing Workflow.

Match Rates GIA

Each comparison of a check image with a check stock image delivers a separate result. This result is then verified against configured minimum match rate for that particular check amount (amount limits).

There are ten different amount threshold that are initially configured by Kofax which range in carrying degrees from AA (a check stock image or other information identified on the document shows a high similarity to that on the stored reference check stock or reference data) to F4 (very little similarity could be found between the comparison images or informations).

Example

Having the key settings (currency: Dollar):

```
LimitGIA-1 = 200000:C1
```

would mean that for up to 200,000 cents (2,000 Dollars) a minimum match rate of at least a C1 is required. Depending on the pre-configured acceptance rate (match rate with amount limit) determines whether to "accept" or "reject" the document and pass it on to the SignCheck workflow process.

These thresholds configured in the **automat2.ini** configuration file under the appropriate SignCheck engine.

Automatic Rule Verification (ARV)

The SignCheck ARV Engine is used to verify the signing rules for visually identified signature(s) automatically.

If a check is routed due to the workflow configuration to the Visual Signature Verification and the operator has selected the signatures, the engine will take the IDs of the visually selected signatures and validate the rules according to the combined ASV process.

The SignCheck Engine configuration file **srvmngr4.ini**, located in the SignPlus directory, enables the configuration of the ARV Engine.

Automatic Pad Signature Verification (APSV)

APSV, also referred to as Teller verification, uses the static and dynamic (pressure, speed) parameters in order to verify a signature at the front office Teller and at Point-of-Sale environment. This verification process provides live verification of a signature and its corresponding signing rules. A signature captured on a pressure sensitive pad is passed to the server via the Java or Thin Clients for Automatic Pad Signature Verification (APSV).

An SDQ server is used in queuing of all the signatures captured from the Java or Thin Clients. The SDQ server and the APSV server use a shared memory **zuAPSV.dll** which enables the management of the queue of the request. This enables for an inter-process communication for the relevant information that is used by the SDQ and APSV servers.

A signature collected from a signature pad is combined with account and amount information and sent as a message to the SDQ server. The SDQ server then places the request in the shared memory for the APSV server to retrieve. The APSV server combines this dynamic information with reference data collected from the SignBase database. The SignCheck APSV Engine processes this bundled information and determines a result. This result is routed back to the Java or Thin Client as a "pass" or "fail".

The diagram below depicts the process of live signature verification:

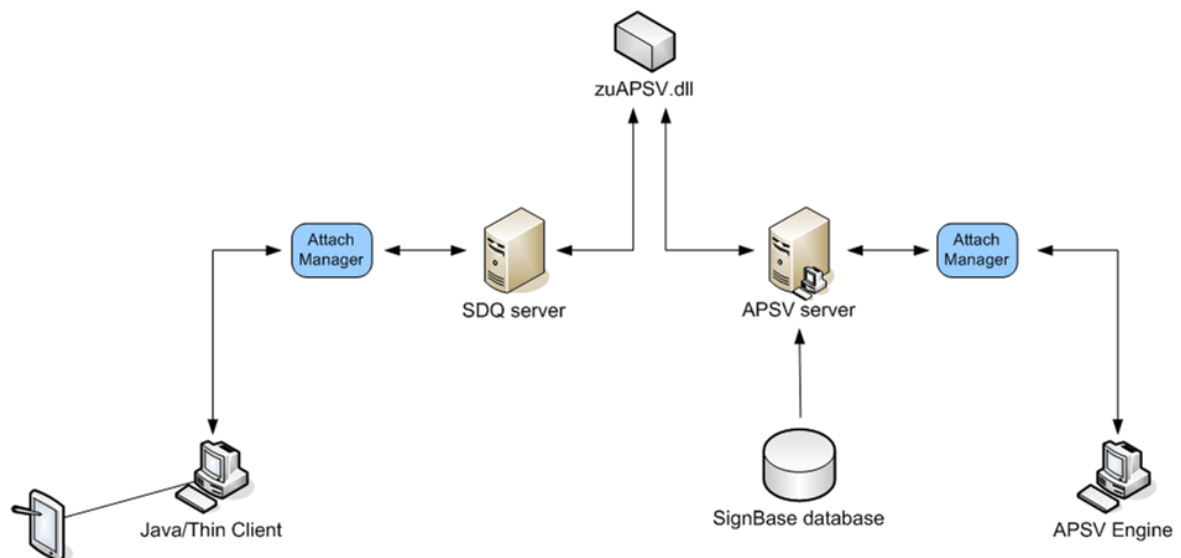


Figure 16: Signature verification with the APSV Engine

The Server Manager configuration file **SrvMngr4.ini** allows for the configuration of the SDQ server and the APSV server. Refer to the [SignCheck Engine Configuration](#) section for a description of the configuration parameters.

SignCheck Engine configuration

The central configuration with the Administration Client enables for the configuration of the SignCheck Engines. It allows for the simple configurations of **Start** and **Stop** times as well as setting the **Trace levels** for the log output.

The **automat2.ini** is the SignCheck Engine specific configuration file which determines how the Engines operate.

General configuration with the Administration Client

Since the automat clients are controlled through the Server Manager, configuration of these clients takes place in central configuration. The following sections describe the configurations that can be done with this file.

ASV Client [ASVCL-1]

In the section [ASVCL-1], the following parameters can be set:

```

Program = automat2.exe ASV
Group = ASV
Port = 2002
Host =
ShowMode = 0
Timeout = 30
RemoteIni = automat2.ini
StatValueWriteInterval = 100
VirtualBufferLimit = 67108864
  
```

```
VirtualBufferStart = 8192  
TcpConAttempts = 3  
TcpWaitTime = 1000  
ConnectionRetryInterval = 30  
ConnectionRetries = 3  
ConnectionRetryAlarm = 3  
Auto2Ex = N  
VerifyEndorsementSignature = N
```

APSV Client [APSVCL-1]

In the section [APSVCL-1], the following parameters can be set:

```
Program = automat2.exe APSV  
Group = APSVCL  
Port = 2008  
Host =  
ShowMode = 0  
Timeout = 30  
RemoteIni = automat2.ini  
StatValueWriteInterval = 100  
VirtualBufferLimit = 67108864  
VirtualBufferStart = 8192  
TcpConAttempts = 3  
TcpWaitTime = 1000  
ConnectionRetryInterval = 30  
ConnectionRetries = 3  
ConnectionRetryAlarm = 3
```

GIA Client [GIACL-1]

In the section [GIACL-1], the following parameters can be set:

```
Program = automat2.exe GIA  
Engine = SAE.dll  
Group = GIACL  
ShowMode = 0  
Host =  
Port = 2009  
Timeout = 30  
VirtualBufferLimit = 12582912  
VirtualBufferStart = 8192  
RemoteIni = automat2.ini  
TcpConAttempts = 3  
TcpWaitTime = 1000  
ConnectionRetryInterval = 30  
ConnectionRetries = 3  
ConnectionRetryAlarm = 3  
StatValueWriteInterval = 100
```


The configuration with automat2.ini

The **automat2.ini** determines the SignCheck Engine operation procedures. These settings are of decisive importance for system performance. These settings are contained in the related FraudOne server that all FraudOne Engines connect to and receive their configuration parameters.

Important When changes are made to this configuration when the overall system is being initially adjusted the Engines must be restarted.

The automat2.ini may only be changed after consulting with your Kofax technical representative.

The following sections describe the configurations that can be done with this file.

[Common]

The following are the parameters for ASV, GIA, and APSV (if installed). All of these parameters can also be specified in the automat (ASV), GIA or APSV section. If a parameter is also specified within the automat, forgery or APSV section, the value defined in the [Common] section will be overwritten.

```

WriteStat = Yes
Interval = 60
NotVerified = Yes
NotLatin = No
RemoveInvalidVariants = Yes
RemovePrefixedDate = No
UseOnlyCleanSnippets = Yes
DontCleanReference = No
SepStepBeside = 15
SepStepUpon = 15
SepMinBorder = 20
FSXResizeImg = 0
NumFsxSplitsBeside = 4
NumFsxSplitsUpon = 4
MinPelsOrder = 400
MinPelsSignature = 400
MinPelsCleanedOrder = 400
MinPelsCleanedSignature = 400
BaseResolutionOrder = 200
BaseResolutionSignature = 200
MinResolution = 150
INICurrency = EUR
MaxSimplicity = 70
FormTypeList = 22, 24
MaxAmount = -1
Limit-1 = 200000: C1
Limit-2 = 1000000: B5
Limit-3 = 10000000: A5
Limit-4 = 100000000: AA
Limit22-1 = 200000: A5
Limit22-2 = 1000000: A3
Limit24-1 = 200000: A5
Limit24-2 = 1000000: A3

```

[Automat]

The [Automat] section specifies the parameters for ASV and ARV only.

The parameters in this section CANNOT be defined in the [Common] section.

```
WriteStat = Yes
SplitSingleSignerAccounts = TRUE
UseDefaultFraud = Yes
Register4Notification = Yes
Port4Notification = 2014
Queue4Notification = 11
Wait4Notification = 5
TwinTestRating = D5
UseRestInst =
CheckCodes =
GiroCodes =
IntlCodes =
AutoRuleVerification = 1
```

[APSV]

The [APSV] section specifies the parameters for Automatic Pad Signature Verification only. If APSV is not installed, this section is not needed.

The parameters in this section CANNOT be defined in the [Common] section.

```
VerifyDynamic = Yes
UseLimits4DynVer = No
UseRestInst = No
CheckCodes =
GiroCodes =
IntlCodes =
```

[GIA]

The [GIA] section specifies the parameters for GIA only.

The parameters in this section CANNOT be defined in the [Common] section.

```
WriteStat = Yes
UseDefaultFraud = Yes
UsePayeeList = Yes
Register4Notification = Yes
Port4Notification = 2014
Queue4Notification = 25
Wait4Notification = 5
FeatureIDs =
LoadRef2Engine = 1
MaskAcctNo = 0000000000000000++++++
MaskSerNo = ++++++
```

[BNO]

Limits (for ASV and GIA) and the corresponding MaxAmount or BNOCurrency parameter can be defined separately for each BNO number.

To do this, individual sections must be defined as follows for each BNO.

Section [BNO-001]

```

MaxAmount = 300000000
BnoCurrency = USD
Limit-1 = 200000: C1
Limit-2 = 1000000: B5
Limit-3 = 10000000: A5
Limit-4 = 100000000: AA
LimitGIA-1 = 90000000: C5
LimitGIA-2 = 950000000: B5
LimitGIA-3 = 5000000000: A5
LimitGIA-4 = 100000000000: AA
FormTypeList = 22, 24
Limit22-1 = 200000: A5
Limit22-2 = 1000000: A3
Limit24-1 = 200000: A5
Limit24-2 = 1000000: A3

```

For additional BNOs, the corresponding section entries are similar. Therefore it is important to know, that a number of BNOs can also be specified as a list of BNOs and as a range of BNOs, e.g. for BNO=002, 007, 008, 009, 010, 012, 015: [BNO-002,007-010,012,015]

Note

- BNOs have to be specified as three-digit-numbers, otherwise they are ignored
- For BNOs for which no special section is defined, the limits defined in the general, block apply
- Values specified in the BNO sections will overwrite the values in the [Common] section.

The [Common] section can contain specific settings such as user access and trace levels for logs. In the event that a key setting is present in both the [Common] section and a BNO specific section, for instance [BNO-008], the entry in the BNO specific section will take precedence over all other settings.

The values shown in these examples are only examples, and are set according to the customer's specification as well as the results of the initial calibration runs, and are only seldom changed after. If changes become necessary, these can only be made in consultation with your Kofax technical representative.

For more information and configuration of the GIA Engines see chapter [GIA Configuration for PreProcessingEngine](#). Refer to the [Related documentation](#) chapter for access to other available Kofax documentation.

SignCheck CRS

The following chapter describes the Combined Risk Scoring in the FraudOne system as well as the configurations that can be performed by an administrator. The availability of certain programs in your installation may vary depending on the products you have purchased.

CRS replaces the static routing of the Workflow component used in pre 3.8 installations to provide a dynamic, configurable workflow. Pre 3.8 workflows need to be rewritten for CRS.

Overview SignCheck CRS

Transaction items pass several different verification steps during processing in FraudOne. That can include different automatic verification steps such as signature or check stock verification, as well as visual (manual) verification steps where users display the item together with associated reference data and decide it.

The SignCheck Workflow and CRS engines will move transaction items from one processing step to the next according to a customer specific defined workflow and decision rules.

System architecture

This chapter gives you a short overview on the CRS architecture.

CRS is shown green, database blue and configuration orange.

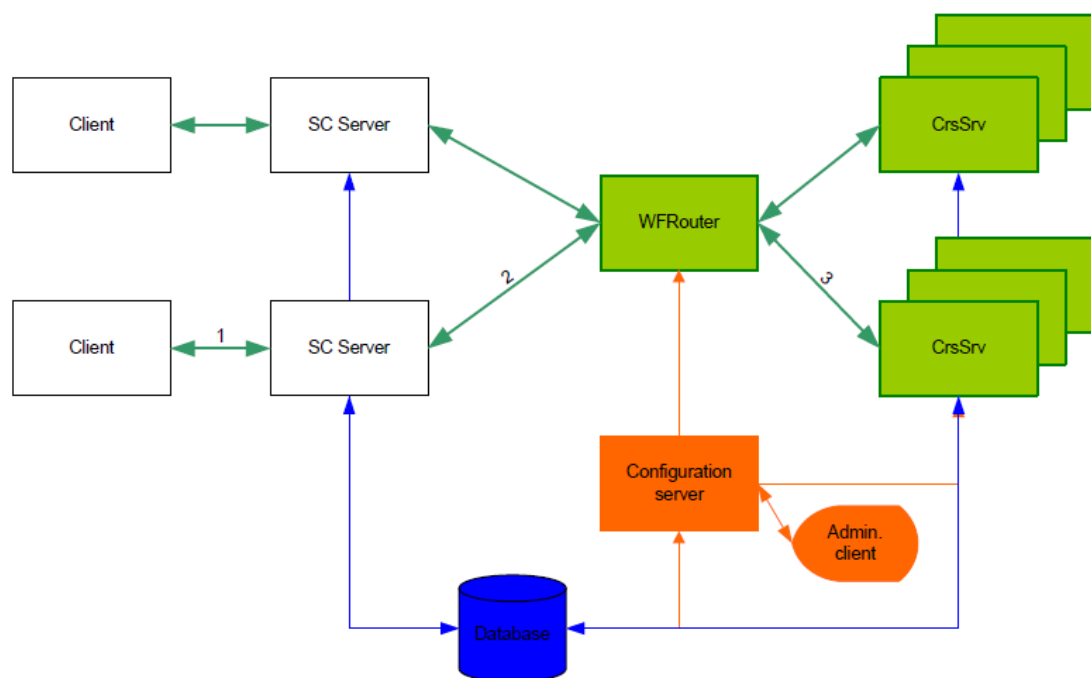


Figure 17: System architecture

One item cycle looks like this:

1. The client (Java Client, Thin Client, verification engine) requests the item to be processed. The server gets the item from the database and presents it to the client. The client makes a decision. The client continues with the next item – the CRS / routing process is asynchronous.
2. The decision is sent via server to the Workflow Router (WFRouter.exe) (Topography entry WorkFlowRouter).
3. The router dispatches the request to the next available CRS engine (CRSSrv.exe) (Topography entry CRSServer). The CRS engine fetches the necessary information to rate

the item from the database. It then applies the CRS rules to the item information and decides what to do with it (computes risk, new queue in workflow, new result). That information is then stored into the database.

The CRS configuration (parameters and rules) is managed by the configuration server. Both WFRouter and CRSSrv read it when starting up. The administration client is used to manage parameters and edit CRS rules.

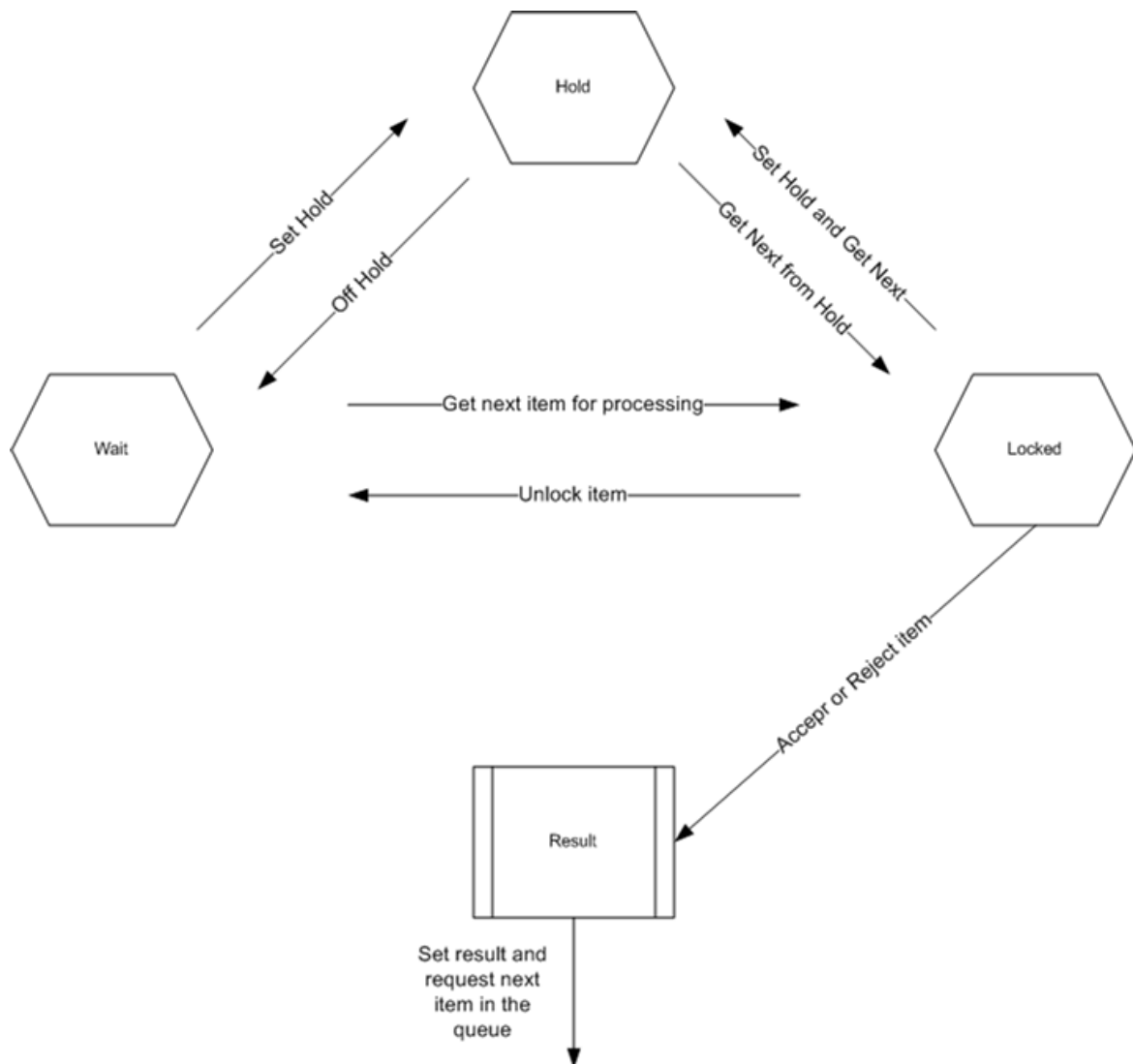
Component	Description
Workflow Router (WFRouter)	<p>Main CRS dispatcher. Exists only once in the system. Accepts CRS requests and routes them to connected CRS engines.</p> <p>The WFRouter manages connected CRS engines from multiple machines and performs load balancing when distributing the requests to available engines.</p>
CRS engine (CRSSrv)	<p>CRS engine that processes individual requests. Retrieves item information from the database, applies the defined CRS rules to it and determines the new target and result of the item. Also writes the client results back to the database.</p> <p>Multiple CRS engines can be run on the same machine and multiple machines can be used as hosts for the CRS engines, since the CRS load balancing procedure does not use attach manager.</p>

Because of these dependencies some care must be taken when starting / stopping components:

- The CRS engine needs to connect to the Workflow Router. It will not be able to work if the Workflow Router is not started or configured. It also need the configuration server to read the parameters and rules to be used.
- The Workflow Router will need the configuration server to retrieve start up information. If the configuration server is not started or set up properly, it will fail to load.
- SignCheck servers, verification servers, Getter and other processes that supply decisions / results will need a started and working CRS system consisting of WFRouter and CRSSrv, since those are responsible for storing the information into the database.

The workflow process

During one processing step a transaction item passes through a series of steps:



The item is waiting to be processed in a queue (**wait**).

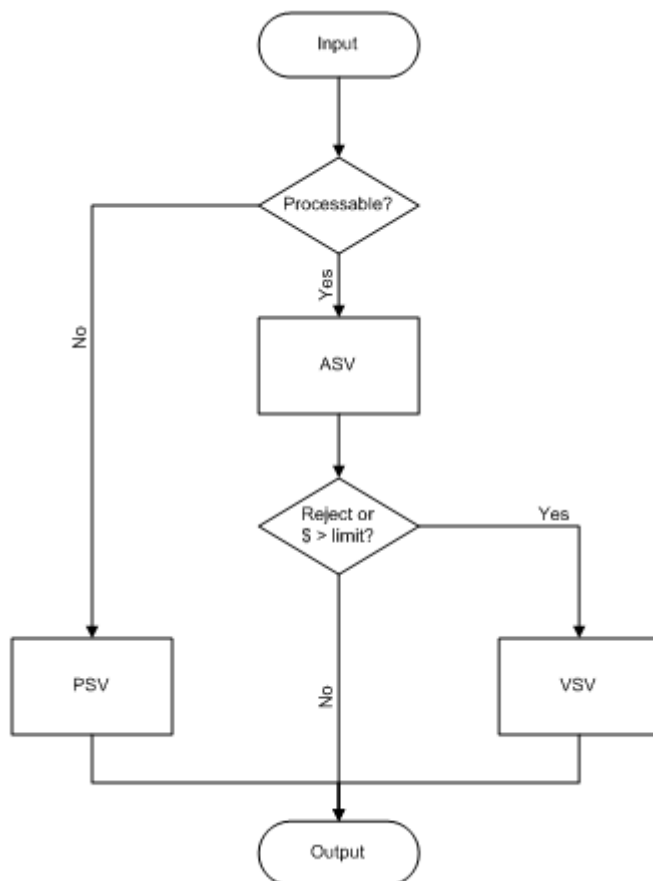
If a user or verification engine requests the item for review it is **locked**. This prevents other users or engines from processing the same item.

The item can then be decided (**result**). The information is passed to the CRS which applies its rules and determines to which queue the item will be moved next. There the whole process begins again.

To prevent users to lock items indefinitely and to resolve possible system errors (visual client crashes or loses connection while the item is locked), locked items will automatically be unlocked by the system after a predefined period of time.

If a user is not able to decide the item, he can put it on **hold**. This prevents the system from unlocking it and assigning it to a different user. The item stays in the same queue until a user (it does not have to be the same one who put it on hold) picks it up and completes processing.

The CRS rules configuration determines which processing steps an item will pass. One sample workflow could look like this:



Non processable items are routed to the physical verification queue (PSV), all others move to automatic signature verification (ASV). Rejected items and items with an amount above a limit will be visually verified by users in the VSV queue.

Using workflow servers

The Workflow or GetNext servers are used in certain installations that have several SignCheck servers accessing the SignCheck database. Each SignCheck server searches the SignCheck database for the next available item. The GetNext server is used as an intermediary between the SignCheck servers and the SignCheck database in order to increase performance. It is used for requesting the next available item for processing. Several GetNext servers can be installed to operate on separate database tables or queue. The GetNext servers are managed using the topographic record entries with the name WFSERVER.

The GetNext servers can be configured for different parameters including the list of servers to use, the GetNext server timeout, and other processing parameters. These parameters are set in configuration file elements with instance type WFSERVER.

Parameter	Description
WFGNCursorTimeout = 60	Time (in seconds) after which cursors in WFGetNext time out. The cursor is rebuilt after this time to insure

Parameter	Description
	<p>priority processing. If no priority processing is desired this time can be set to a higher value.</p> <p>Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.</p> <p>Default: 60</p>
<code>WFCheckUserInGetNext = 0</code>	<p>Does WFGetNext have to check if the user has not processed the document returned in one of the previous queues?</p> <p>Use with care, since WFGetNext performance decreases. The preferred way to do this is to disable users working on different levels via user permissions.</p> <p>Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.</p> <p>Possible values: 0, 1</p> <p>Default: 0 = FALSE</p>
<code>WFGetNextServers =</code> <code>0:computer.domain.com:2017</code> <code>,11:computer.domain.com:20</code> <code>18</code>	<p>The list of servers to use for WFGetNext (if needed). Format is:</p> <p style="padding-left: 40px;"><code>queue:servername:port,queue:servername:port,...</code></p> <p>If this is not set, classic WFGetNext with direct DB access is used.</p> <p>Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.</p>
<code>WFSocketTimeout = 1000</code>	<p>If using GetNext servers, the timeout (in milliseconds) after which WFGetNext gives up. GetNext will retry three times and resend the message another three times. Thus the total timeout will be $3 * 3 * \text{WFSocketTimeout}$.</p> <p>Default: 1000</p> <p>Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.</p>
<code>WFGNSErrorTimeout = 0</code>	<p>1 - Means that WFGetNext will fall back to using direct DB access if the GetNext server can't be reached. This will continue for WFGNSErrorTimeout seconds, after which WFGetNext will try to reach the server again.</p> <p>0 - Means that no fallback method is used.</p> <p>Default: 0</p>

Note The values in this table can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the **[Common]** section.

Using CRS

Starting and stopping components

Both the Workflow Router (WFRouter) and CRS (CRSSrv) are controlled by Server Manager (SrvMngr4). After Server Manager start the processes will be started automatically if specified in the topography record. Starting and stopping the components can be done via Server Monitor or SNMP.

Be aware that following dependencies exist:

- SignCheck servers need a running WFRouter to provide queue information.
- WFRouter needs a running configuration server (SignBase server) to supply configuration information during startup.
- CRSSrv needs both a running configuration server and a running WFRouter to connect to.

If the above conditions are not met, WFRouter and/or CRSSrv will display an error and turn red in the Server Monitor display. After the error condition has been corrected (configuration server started, WFRouter started), the components will resume normal operation and turn green.

Supporting tools and utilities

WFFlush

The Workflow Flush utility is used to flush all active documents from the SignCheck workflow queues. All documents that have not yet completed the processing cycle and match the Workflow Flush selection criteria will be moved to the Output queue and an "Unprocessed" final result code will be written.

WFFlush is normally started from the command line and is not present in the topographic record in that case. It can be configured in the topographic record and so then started from the Server Monitor, for example.

Important The Workflow flush process is irreversible and should only be used in extreme and specific cases. This process will also generate "Error Messages" with any running FraudOne visual and non-visual clients.

When starting the WFFlush application the following parameters need to be entered:

Parameter	Description
Database	The name of the database
Server	Server name (will only be used by some database systems)
Userid	User id for database access
Password	Password for database access
BankNo	BNO to work on. The default is 'all', which means that all BNOs will be processed.

Parameter	Description
Date	Date after which documents get flushed (use YYYY-MM-DD format). The default is 'all', which means that all data will be processed.
DoNotAsk-IKnowWhatIAmDoing	Don't ask for confirmation. Set this only if it is really necessary.

Some or all of the above can be configured in **WFFlush.ini** file element from the database. Parameters configured here will not be queried from the user. Settings in the INI file need to be done in the **[INPUT]** section.

You can specify one or more INI files on the command line, they will be processed sequentially. This is useful if you want to configure WFFlush to start at specific times and work automatically.

WFDecide

WFDecide provides a way to automate decisions by implementing a 'cut-off' function. Items (documents and images) in a queue that have not been decided on by specific time are automatically decided with the **WFDecide**.

WFDecide is started by the Server Manager via a StartTrigger that is specified in the configuration file element for the WFDecide program instance(s). These have instance type WFDecide and the topographic record element WFDecide controls which WFDecide instances can be started.

The configuration file element includes the sections that need to be worked on by the WFDecide. Each section contains the settings that are used to filter the documents to be processed. This contains the queue number, BNO, branch number and hold status. WFDecide requests the next document available that matches the filtering criteria for that section.

Important Section ordering is important. When WFDecide changes a document, the Workflow Router selects it and routes it to a different queue. If you also want to process that second queue and need to catch the document routed to it, the section describing the second queue must follow the section that decided the first. Otherwise, the document will be routed to the second queue without being decided by WFDecide.

The WFDecide gets the next document that is sorted by workflow priority. This document is decided according to the result settings:

- If LastResult is set to 0 then the previous result is used
- If LastResult is not set to 0 then the result is taken from NewResult

After a section is finished, WFDecide continues with the next one. After all sections are processed, WFDecide will either stop, or continue running and start over, depending on the StopWhenReady setting, which is also configured in the configuration file element.

If no more documents are found, WFDecide will continue to wait for the time given in IdleTime for that section before proceeding to the next one. The reason for this is that when processing multiple sections, documents from a previous section can get routed to the current section by the workflow. IdleTime needs to be chosen carefully in order to give the workflow enough time to route all documents that have been decided in previous sections.

If WFDecide finds a new document, IdleTime is counted again from zero. After IdleTime expires, WFDecide continues with the next section.

Installation requirements

WFDecide requires the workflow ZuWF.dll to be located in the same directory. It runs under the Server Manager, which support the StartTrigger setting.

SrvMgr4.ini settings

The configuration file element for the **SrvMgr4.ini** should contain the database related settings (Database, Userid, Password).

If you start WFDecide via StartTrigger make sure that the topographic record for the WFDecide allows it to start automatically. It is safe to set a StopTrigger a couple of minutes before the value in StartTrigger.

A sample configuration file element for the WFDecide instance type is described below:

```
Program=WFDecide.exe
Group=Cut-off
ShowMode=0
ConfigFile=.\WFDecide.ini
StartTrigger=16:00
```

WFDecide.ini

The WFDecide.ini file contains following configuration parameters:

[General] section

```
DocCheckWaitTime =
SanityCheckRefresh =
StopWhenReady =
```

[WFDEC] section

```
Queue =
OnHold =
BNno =
BranchNo =
LastResult =
NewResult =
IdleTime =
```

CRS configuration

Workflow Router configuration

This section describes the configuration of the SignCheck Workflow Router.

Configuration for the Server Manager

The following parameters can be set in the configuration file element for the instance type WorkFlowRouter:

```
Userid = user
Password = password
```

```
Database = SignPlus
Port = 2014
WFPort = 2018
CRSPort = 2019
ConfigFile = WFRouter.ini
CRSRuleFile = crs.xml
CRSDtdFile = crs.dtd
```

WFRouter.ini configuration

The configuration file element with the file name given in the ConfigFile property is also stored in the database and referenced also with the instance type WorkflowRouter.

The file element is a complete configuration file containing sections that are used by the Workflow Router. The parameters in the sections are described below.

Important Consult your Kofax Support Representative before performing other configurations that the ones described below.

[General] section

This section contains the general configuration parameters:

```
LockTimeout = 300
PriorityBoostTimeout = 86400
PriorityBoostRefresh = 86400
SanityCheckRefresh = 600
CommitCount = 1
UsePriority = 1
```

CRS configuration

This section describes the system configuration of the CRS component. The routing (workflow and decision rules) that can be changed by the customer is described in a separate document.

Configuration in the Server Manager configuration file

The following parameters can be set in the configuration file element for the instance type CRSServer:

```
CRSConfigFile = crs.ini
```

CRS.ini configuration (Configuration Server)

The file name given in the CRSConfigFile parameter together with the instance type CRSServer is mapped to a configuration file element which is itself a complete INI file containing multiple sections as follows:

Section [Parameters]

```
RuleFile = crs.xml
DtdFile = crs.dtd
```

```

WFServer = localhost
CRSPort = 2019
RiskThreshold = 0
VipMultiplier = 1

```

Section [Constants] - replaces pre 3.8 Limits

Parameter	Description
x = y	Constants definitions as specified in the CRS rules. The [Constants] section contains the default values, those can be overwritten by BNO dependent values below.

Section [BNO-???] - replaces pre 3.8 Limits

Parameter	Description
x = y	BNO specific constant values that override the default values above. BNO values are set in the section name ([BNO-019]).

Workflow Server configuration

Configuration in the Server Manager configuration file

The following parameters can be set in the SrvMngr4.ini configuration file element for the instance type WFServer:

```

Userid = user
Password = password
Database = SignPlus
WFSrvPort = 2017
WFSrvErrTimeout = 5

```

Workflow API Configuration

The following table describes the SignPlus Workflow API configuration parameters. These parameters are available for all SignPlus components that use the Workflow API (SignCheck server, ASV server, etc.).

The parameters can be set in the configuration file elements for file name **SrvMngr4.ini**:

Parameter	Description
Name = Monitor1	Log pipe name Default: Monitor1
WFGNCursorTimeout = 60	GetNext cursor timeout (in seconds). Even if the cursor used still has entries it will be refreshed after this timeout. Default: 60

Parameter	Description
WFCheckUserInGetNext = 0	Does the system need to check that the same user can't decide the same item twice? Possible values: 0, 1 Default: 0
WFRouterAddr = localhost	The name of the machine the workflow router is running on. Default: localhost
WFRouterPort = 2018	The port number used to send messages to the workflow router. Default: 2018
WFSocketTimeout = 1000	The timeout (in milliseconds) used when communicating with the workflow router. Default: 1000
WFGetNextServers =	List of GetNext servers to use in the format: queue:server:port,queue:server:port,... Empty if no GetNext servers are used. Default: no set
WFGNSErrorTimeout = 0	GetNext server error timeout before fallback to normal (local) GetNext. Default: 0

CRS Variables

CRS predefined variables

Following variables are available in any CRS installation:

Location	IF (fixed)	Variable	Type	Source
SignBase	111	Account type	Long	CUSTOMERTYPE in CUSTOMER table
	112	BNO	String	BNO in CUSTOMER table
	121	Branch number	String	BRANCH_CODE in ACCOUNT table
	122	ZIP code	String	new field DEMOGRAPHIC_LOCN in ACCOUNT table
	123	Age of the account	Long	Calculated by the DBHelper process using the new 'account opening' date

Location	IF (fixed)	Variable	Type	Source
	141	Age of matched signature	Long	Calculated by the DBHelper process using DATESCANED of the matched signature
	113	Customer since	Long	Calculated by the DBHelper process using the new 'Customer since' date
	131	Age of matched signatory	Long	Calculated by the DBHelper process using the signatory's BIRTHDATE
	114	VIP	Boolean	Retrieved by the DBHelper process using the customers VIP field
		Account extensions (?)	(?)	Extension1 table
Item (SC)	1	Last Queue	Integer	a queue number, used by the CRS engine to check the last queue the item was routed to
	211	Amount	Double	AMOUNT_LOCAL in SC_INTERFACE table
	212	Document type	String	FORM_TEXT_CODE in SC_INTERFACE table
	213	Correction item	Boolean	Check if FORM_TYPE='3' in SC_INTERFACE
	214	IRD	Boolean	Check if FORM_TYPE='4' in SC_INTERFACE
		Extensions (?)	(?)	Customer defined tables
Result (SC)	311	ASV result	Long	Document based RESULT in SC_RESULT where feature id=769
	312	ASV match rate	Long	Document based MATCH in SC_RESULT where feature id=769
	321	APIA result	Long	Document based RESULT in SC_RESULT where feature id=1025
	322	APIA match rate	Long	Document based MATCH in SC_RESULT where feature id=1025
	331	PAD	Boolean	Document based RESULT=1 in SC_RESULT where feature id=513
	332	PAD blacklist	Boolean	Document based RESULT=1 in SC_RESULT where feature id=515

Location	IF (fixed)	Variable	Type	Source
	333	PAD payee match	Boolean	Document based RESULT=0 in SC_RESULT where feature id=514
		Getter results	Boolean	Document based RESULT in SC_RESULT where feature id between 257 and 265

Predefined variables can be enabled / disabled in the rules editor in order to clear the display of unneeded information.

CRS customer specific variable configuration

CRS rules definitions rely on the presence of a predefined list of "variables" that allow the rules programmer to access information from the SignBase and SignCheck database during rules processing. The standard set of variables (described in [CRS Predefined Variables](#)) can be extended to include additional Result and Match Rate items from the SignCheck processing engines and visual queues and additional data items from customer specific SignBase extension tables.

These additional variables are configured by means of one or more configuration file elements in the central configuration database.

The information is stored under the file name **CRSDBHelper.ini** which must contain a section name **[VariableList]** containing the keys and values described below. The address mapping used to access this file uses the instance type of the calling program; since several programs can access this file, it is recommended that the address map record contains a "*" for the instance type.

[VariableList]

This imbedded section contains the definitions of the customer specific additional CRS Rules variables.

Parameter	Description
<code>VariableCount = 0</code>	<p>The number of additional variables defined in the list.</p> <p>The default is 0 also when no CRSDBHelper.ini can be found in the Configuration Database tables.</p> <p>Note that the variable definitions (see below) must be numbered sequentially from 1 to the number given in this configuration item. This numbering completes the Key Name by replacing the "<n>" with the specific variable number.</p>
<code>Var.<n>.Name = name</code>	<p>The name of the variable.</p> <p>This name is used only for display in the rules editor.</p> <p>Example</p> <p><code>Var.1.Name = Customer Married Since</code></p>
<code>Var.<n>.Type = type</code>	The data type of the variable

Parameter	Description
	<p>Supported values are: int, double, bool, string</p> <p>Example</p> <pre>Var.27.Type = double</pre>
Var.<n>.VarId = idnum	<p>A unique variable identifier number.</p> <p>Numbers in the range 30000-30999 are available for customer use without reference to Kofax.</p> <p>Example</p> <pre>Var.12.VarId = 30211</pre> <p>Kofax reserves the right to use numbers outside this range at any time and for any purpose without further communication with the customer.</p> <p>Important Customers using numbers outside this range run the risk of causing unspecified system failures.</p>
Var.<n>.Object	<p>The database object type containing the data referred to by this variable.</p> <p>Useful values are: random, result, extension0, extension1, extension2, extensionb</p> <p>random - defines a variable that will contain a random number in the range defined by RangeLow and RangeHigh below.</p> <p>result - defines a variable that can contain either a CRS Result value or a Match Rate value from a specific feature code.</p> <p>extension0 - defines a variable that can contain a data item from the CUSTOMER EXTENSION table.</p> <p>extension1 - defines a variable that can contain a data item from the ACCOUNT EXTENSION table.</p> <p>extension2 - defines a variable that can contain a data item from the SIGNATORY EXTENSION table.</p> <p>extensionb - defines a variable that can contain a data item from the ACCOUNT_IMAGE EXTENSION table.</p>
Var.<n>.Link	<p>For Object=result variables, this value identifies the Feature Code for which the result or match rate is required.</p> <p>For Object=extension2,-b variables, this value identifies the feature code to be used to select the matching signatory or account image.</p>
Var.<n>.Match	<p>For Object=extension2,-b variables, this value identifies which of the two possible matching object fields from the Feature Code record contains the SIGNO or IMAGENO.</p>
Var.<n>.FieldId	<p>For Object=result variables, this value selects either the Result (value 1) or the Match Rate (value 2) stored in the database for the Feature Code.</p> <p>For Object=extension0,-1,-2,-b variables, this value identifies the specific extension field from the corresponding table.</p>

Parameter	Description
<code>Var.<n>.RangeLow</code>	For Type=int or Type=double variables, this value, if present, represents the minimum permitted value of the variable. This only affects the Rules Editor; the variable value itself is not checked against this range.
<code>Var.<n>.RangeHigh</code>	For Type=int or Type=double variables, this value, if present, represents the maximum permitted value of the variable. This only affects the Rules Editor; the variable value itself is not checked against this range.

Kofax FraudOne Reports

The following chapter describes the Kofax FraudOne Reports Server component. It uses the data warehouse repository to provide system reports for previous processing days.

Overview Kofax FraudOne Reports

The reports are generated with Kofax Insight reporting and dashboard application, that provides reporting functionality for past processing days. It uses data provided by the data warehouse database to format text and graphic reports that can be accessed and printed from a web browser.

Reports are organized in groups:

- SignBase: reports on reference data
- SignCheck: reports on check processing
- Service: reports on data loading and service tasks

In addition to the core groups above, extended reports could be requested or customer specific groups can be created on request.

System Architecture of the Kofax FraudOne Reports

The Kofax FraudOne Reports application is running independently from other Kofax FraudOne components (not under Server Manager control) and can be started separately. It accesses the data warehouse database directly.

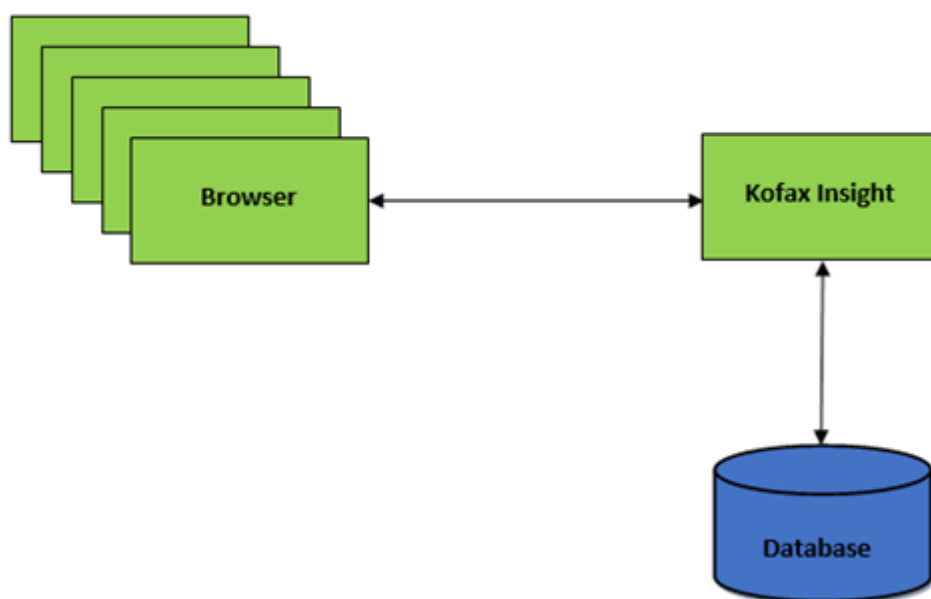
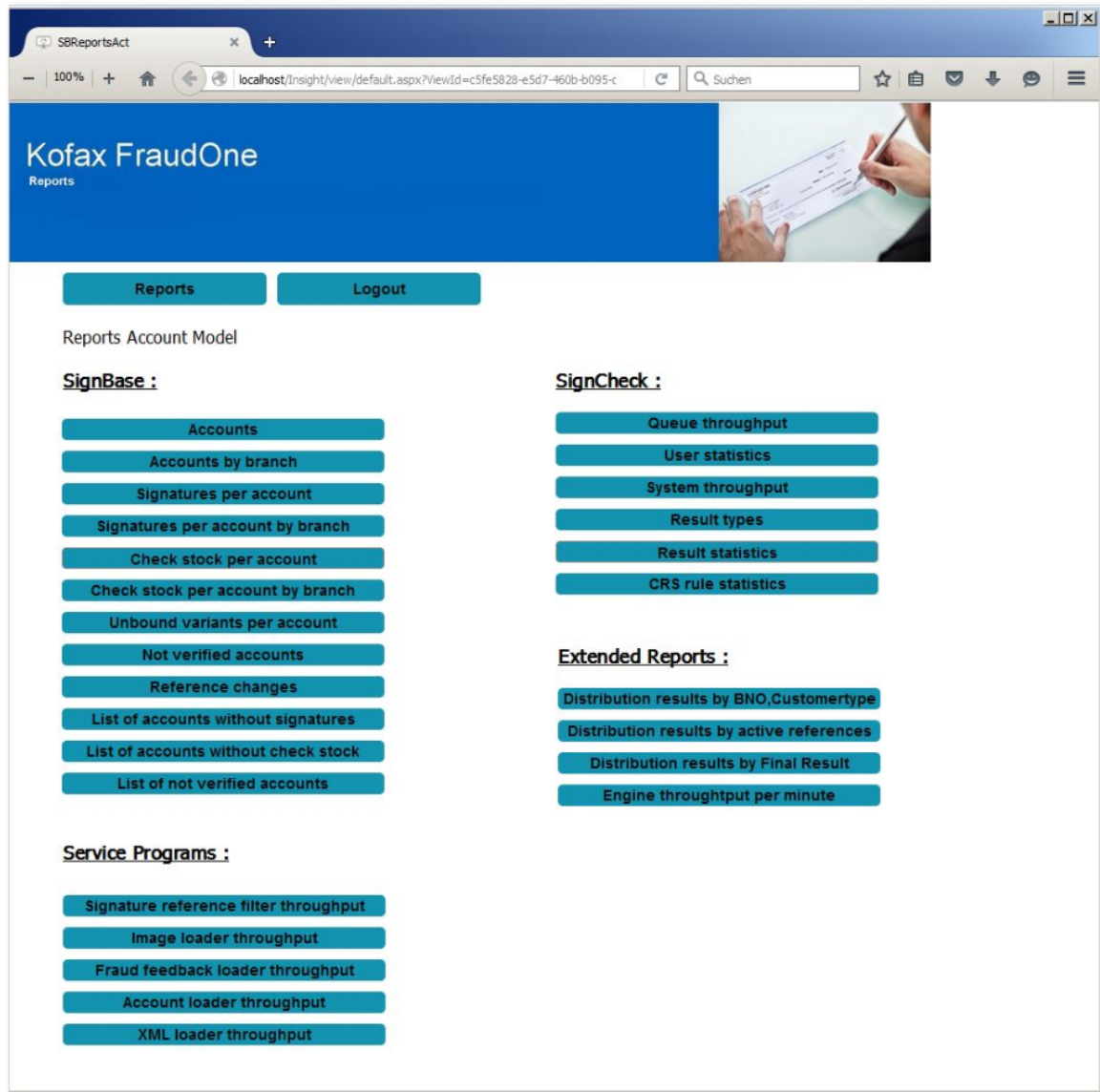


Figure 18: System architecture of the Kofax FraudOne Reports

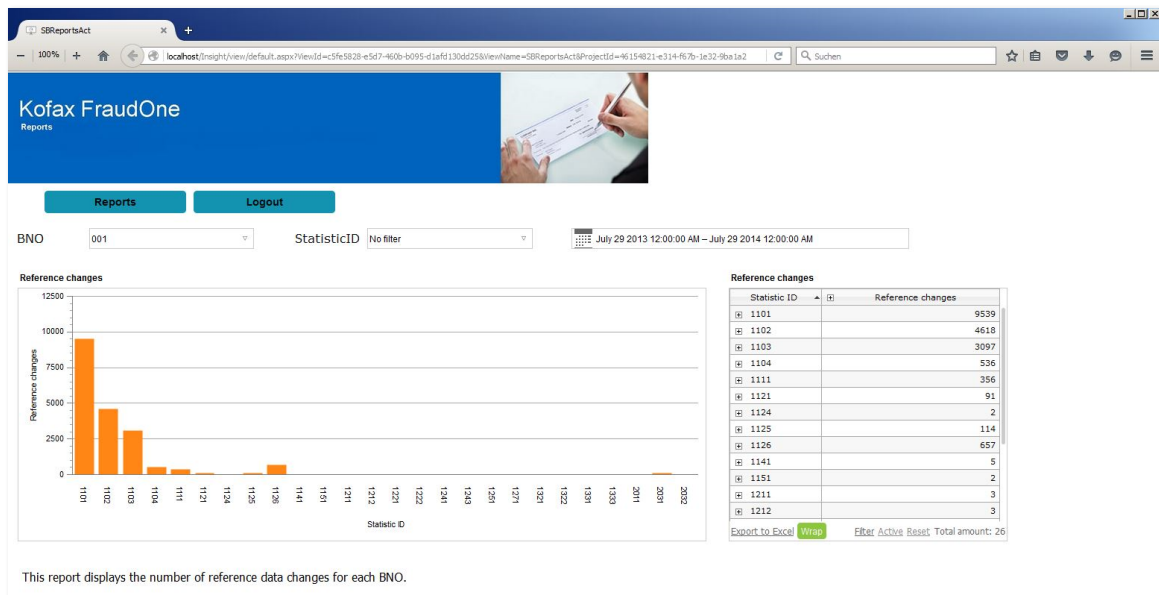
Reports usage

Reports are accessed via a web browser. To be able to access reports, a user must log in. He will be able to see reports for all BNOs.



The menu will show the report groups and allow the user to select a report. The standard delivery does not include the group **Extended Reports**. The **Extended Reports** can be purchased separately.

The selection of a report is done by double click on the reports to display. Clicking on the **Reports** button will return to the menu page. Clicking on the **Logout** button will logout from Kofax FraudOne Insight.



Reports can be printed using the browser print functionality. The print layout differs from the screen layout.

Also the user can export the displayed table as Excel document.

Kofax FraudOne Reports installation

A simplified installation package has been created for the standard Kofax modules to be used on the Windows operating system.

The following paragraphs will describe the installation steps if you use this package.

Packages

The Kofax FraudOne Report Server component can consist of up to two packages:

- **Kofax Insight 5.2.x** - the Insight application that contains the installation program and Insight documentation needed for the Kofax FraudOne Reports.
- **Reports** - the Kofax FraudOne Reports.

Prerequisites

Database drivers for the database used need to be installed on the machine. Use one of the drivers supported.

Installation

Kofax Insight Application

The Kofax Insight application will be installed by installation program. See **KofaxInsightInstallationGuide_en.pdf**.

Kofax FraudOne Reports

The Kofax FraudOne Reports will be installed by importing the reports. See the *Kofax FraudOne Installation and Migration Guide*.

Server Monitor

Overview Server Monitor

The FraudOne Server Monitor acts as a control interface for all server side systems in the FraudOne product suite. The Server Monitor is used to start processes, monitor the status of all FraudOne server processes, and create log files remotely for the server processes.

The Server Monitor enables the administration of the following:

- FraudOne application server processes
- FraudOne verification processes managed by Server Manager instances
- FraudOne Service Programs through the network

The Server Monitor can be used to perform the following actions:

- Connect and disconnect to server computers
- Start and stop processes
- View server processes
- Creation and filtering of log files

The following section details the setup and simple configuration of the FraudOne Server Monitor.

Server Monitor setup

All machines where the Server Monitor is to be used must be configured as a member of the FraudOne installation using the **SetupCfg.cmd** utility that is installed as part of the Server Monitor installation.

The **SetupCfg.cmd** utility has a number of options. The options that are needed for a Server Monitor installation are any of the "-init" commands and "-addserver" (or "-addauthserver" if authentication is done by a different server to the configuration).

Example

```
SetupCfg.cmd -initremote leron
SetupCfg.cmd -addserver okalani.emea.kofax.com:2000
SetupCfg.cmd -addauthserver ginevra.emea.kofax.com:2000
```

Note Use the keyboard for entering the code snippets, in case the Copy/Paste functionality does not work correctly.

The name used in the "-initremote" option must be an installation unique name (20 characters or less) that identifies the machine where the Server Monitor is installed.

The name used in "-addauthserver" or "-addserver" must be the network address of a system where the SignBase authentication server is installed and running, including the configured port for the server.

Use "-addauthserver" if the installation is configured with separate server instances for user management and central configuration and "-addserver" alone if these roles are assigned to a single server instance.

The Server Manager runs on the server computer and controls the server processes. The Server Monitor requires the following connections in order to communicate with the Server Manager:

- UDP based connection for passing commands to the manager process. UDP is used to minimize network traffic impact of the monitoring process
- A TCP based connection for transmitting log entries. This connection is only active while logging is on

The port numbers should both be configured to be the same for the Server Manager and Server Monitor. The port number for the Server Manager can be specified in the configuration file element for file name **srvmngr4.ini** and instance type ServerManager:

Settings		Default	Description
MonRequestPort	CmdPort	2004	UDP port used to pass commands
MonLogPort	LogPort	2005	TCP port for log information

Using the Server Monitor

Using the embedded Help feature

The Help folder enables for an intuitive assistance when using the Server Monitor. The **Help** menu allows for access to the Help folder and a helpful Tutorial. The Help and Tutorial is also available in document format and can be downloaded from the Kofax document database. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Button bars and screen explanations

The Server Monitor enables actions to be performed by using the standard windows pull-down menu bar, a button bar, or by right-clicking a computer process and selecting the desired action. The Server Monitor consists of a double-pane layout. The left pane allows for the administration of computer server and processes. Using a tree representation, all components (server processes, verification processes, etc.) are attached and can be grouped. The right pane displays one or more Log windows according to the user's needs. The Logs can be configured to use defined output filters.

The following depicts the Server Monitor GUI:

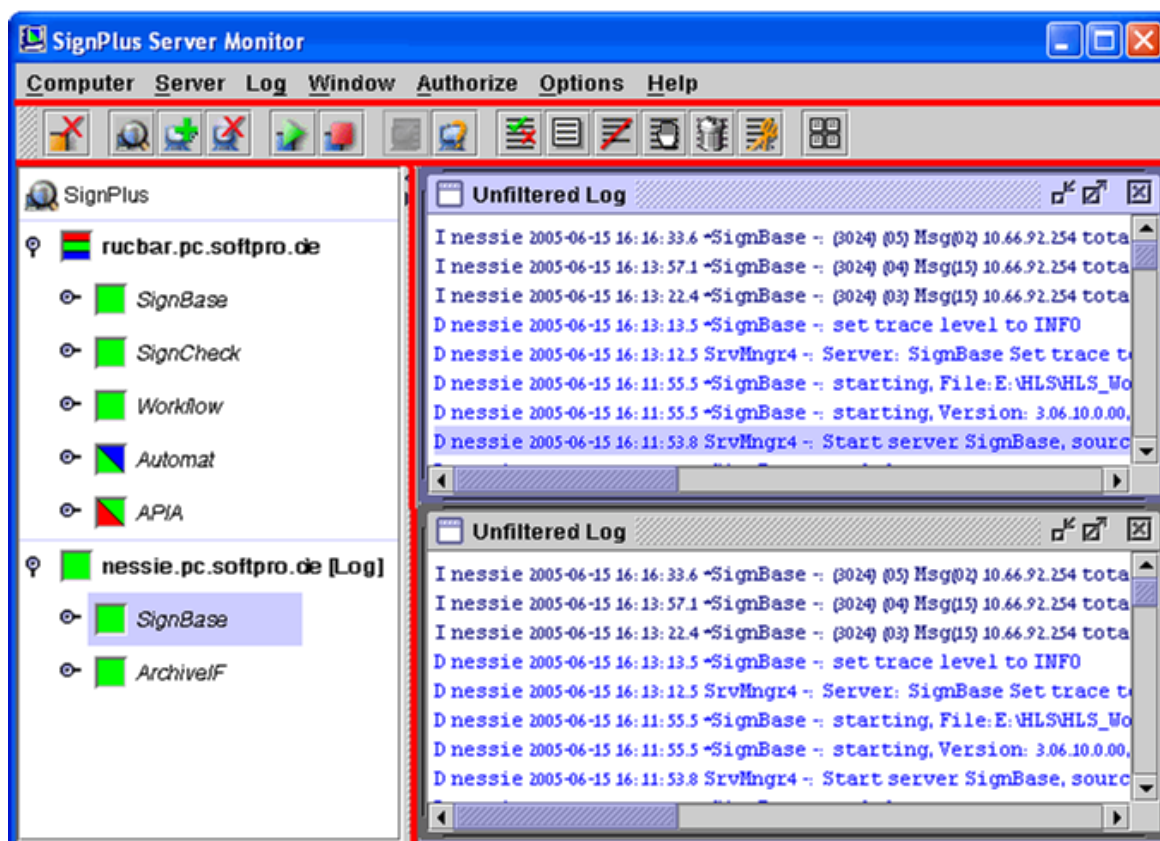


Figure 19: Server Monitor GUI

Authorization

The server monitor prompts for a user-id and password when it starts. This user-id must be present in the SignBase user management database and have the SrvMon_Admin access right.

Server Monitor configuration

The configuration file element **SrvMon.properties** for instance type ServerMonitor enables the configuration of the Server Monitor.

Parameter	Description
DiscoverBroadcast	When trying to find managed computers, the monitor will use this broadcast network address to discover computers where the Server Manager runs on. You can restrict the search by specifying the broadcast address of your local subnet. The search will only work on your local network. Default: 255.255.255.255

Parameter	Description
DiscoverFrom DiscoverTo	<p>If broadcasts won't work for you (the managed computers are on a different network), you can search the computers one by one. In this case you have to leave 'DiscoverBroadcast' empty and use 'DiscoverFrom' and 'DiscoverTo' instead. Type in the start IP address of your search range into 'DiscoverFrom' and the end IP address of the range into 'DiscoverTo'. The Monitor will now search this range for managed computers.</p> <p>Beware that this method is much slower than using broadcasts.</p> <p>Default: None</p>
CmdPort	<p>The Monitor will send commands to the Server Manager using the UDP protocol and the port number specified in 'CmdPort'.</p> <p>If this port number is in use on your network, specify a different one here. Be aware that this port number must match the one configured in the Server Manager for the two components to be able to talk to each other.</p> <p>Default: 2004</p>
LogPort	<p>The Monitor will exchange log information with the Server Manager using the TCP protocol and the port number specified in 'LogPort'.</p> <p>If this port number is in use on your network, specify a different one here. Be aware that this port number must match the one configured in the Server Manager for the two components to be able to talk to each other.</p> <p>Default: 2005</p>
LocalAddress	<p>If you have installed the Monitor on a multi-homed host (one with more than one network card or address), you can sometimes have problems with the system identifying the correct address to use when sending commands.</p> <p>The symptoms to this are, that the monitor displays the information from the managed computer, but reports an authentication error when trying to send commands, even though the user id and password is correct (see Authentication).</p> <p>You can get around this problem by typing the IP address to use into the 'LocalAddress' field. This will have to be one of the IP addresses of your computer, specifically the one of the interface pointing to the network where the managed computers sits.</p> <p>Use this field only if the Monitor does not work otherwise.</p> <p>Default: Empty</p>
MaxLogLines	<p>When displaying log messages in the log window, the Monitor will only show you a specific number of messages. After the maximum number of messages has been reached, each new message received will delete the oldest one.</p> <p>You can configure the number of lines to be shown with the 'MaxLogLines' parameter.</p> <p>Default: 2048</p>

FraudOne SNMP

This chapter describes the FraudOne SNMP feature as well as the installation procedures. The SNMP product and its features are only available if purchased.

Overview FraudOne SNMP

It is possible to manage the FraudOne server modules and verification processes over the network using external network management tools featuring standardized SNMP (Simple Network Management Protocol). SNMP uses a set of message protocols in order to manage complex networks. For this purpose an SNMP client has to be installed on the managed FraudOne server. SNMP is an optional feature to be used with the FraudOne Server Monitor.

SNMP is an alternative way of controlling the different FraudOne components by integrating the FraudOne Server Manager into the network management system.

At this time, SNMP Support is only available for FraudOne Servers running under Windows. SNMP support for FraudOne components is provided in the form of an extension module for the Windows SNMP agent.

The following describes the SNMP facility of the FraudOne system:

- The SNMP manager sends requests to the Windows SNMP agent running on the machine where the FraudOne servers are installed.
- The Windows SNMP agent loads the FraudOne extension module at startup and forwards MIB (Management Information Base) related requests to the FraudOne SNMP module.
- The FraudOne Server Manager contacts the FraudOne SNMP extension module at startup, if available. The extension module then forwards incoming SNMP requests to the Server Manager.

The following diagram depicts the SNMP module in the FraudOne system:

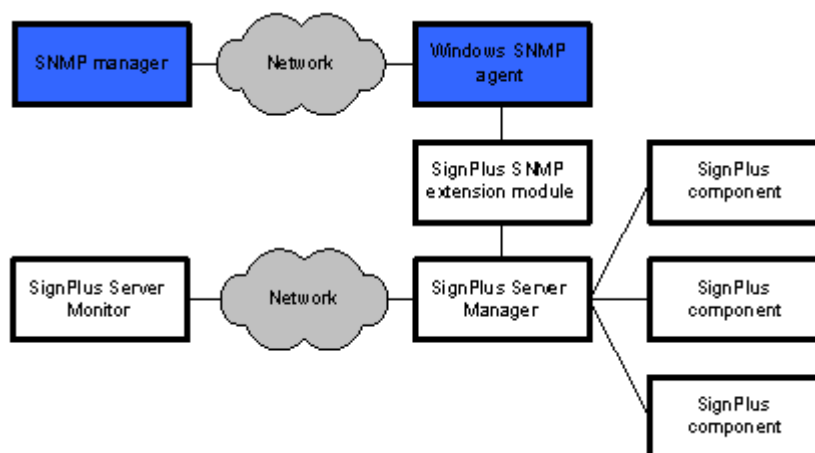


Figure 20: SNMP manager in the FraudOne system

Files used

The SNMP module requires the following files:

- **SpSrvMonSNMP.dll**
FraudOne Server Manager SNMP extension to the Windows agent.
- **SpSrvMon.mib**
FraudOne MIB description for SNMP management console use.
- **SpSrvMonSNMP.reg**
Registry information for the SNMP extension agent (to be replaced by the FraudOne installation tool).

SNMP installation

The following describes the installation procedures of the SNMP module in the FraudOne system:

- Make sure the SNMP service is installed. Consult your Windows documentation for details.
- Install the **SpSrvMonSNMP.dll** in the desired directory, preferably in the same directory as the Server Manager.
- Edit the **SpSrvMonSNMP.reg** file and replace the drive and directory location for **SpSrvMonSNMP.dll** with the path you installed it to in the previous step.
- Input this information into the registry with **regedit SpSrvMonSNMP.reg**. This can also be inputted manually.
- Configure SNMP information, authorization and trap destinations (consult your Windows documentation for more details).
- Stop and then restart the SNMP service via the Windows service manager.

SNMP information stored in the Windows registry:

Registry key	Value	Description
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ExtensionAgents	"SpSrvMon"="SOFTWARE\SOFTPRO\SignPlusSNMP\CurrentVersion"	This registry entry contains a list of SNMP extension modules the Windows SNMP agent will load on startup. We insert an entry here containing the name of the registry key where the software is registered.
HKEY_LOCAL_MACHINE\SOFTWARE\SOFTPRO\SignPlusSNMP\CurrentVersion	"PathName"="c:\signplus\SpSrvMonSNMP.dll"	Registry key under which the software is registered (see above). It contains the information where the module can be found on disc. You'll have to modify this to reflect the place where the SNMP module is installed.

SNMP objects description

The available SNMP objects are described in the **SpSrvMon.mib** SNMP MIB definition file. This file can be imported into the SNMP manager to enable the use of this information.

Using the SNMP Module

The following are the SNMP usage considerations:

- In order to manage the FraudOne servers via SNMP, the SNMP agent service has to be started before starting the FraudOne Server Manager.
- If the Windows SNMP agent is stopped and restarted then the FraudOne Server Manager also needs to be restated.
- FraudOne messages are sent via SNMP traps to the trap destinations configured in the Windows SNMP agent.
- The server status and trace level can be queried and set via the SNMP commands.
- It is best to use the SNMP agent to control server tasks via the Server Manager.
- SNMP agent works in parallel to the Server Monitor. The Server Manager can be controlled simultaneously via the SNMP and the Server Monitor.
- Security access can be controlled via the SNMP security. Refer to the chapter SNMP which describes the installation and usage of the SNMP security feature.

FraudOne Service Programs

General

This chapter describes the FraudOne Service Programs in the standard functionality as well as the configurations that can be performed by an administrator. The availability of certain service programs in your installation may vary depending on the products you have purchased.

FraudOne Service Programs requirements

- All Service Programs require an installed Java Runtime environment. See the *Kofax FraudOne Installation and Migration Guide* regarding proper installation procedures.
- The Signature Reference Filter (SRF) and XML-Loader (XL) Service Programs require a TCP/IP connection to a SignBase server.
- Account Loader (AL), Image Loader (IL), Getter, Putter and Day's Final Processing (DFP) Service Programs require a connection to the SignBase and SignCheck database and its associated JDBC classes.
- If a mono (black & white) signature cleaning is used, the Account Loader and Getter Service Programs require an updated FraudOne licensing key. Consult your Kofax business representative for further information regarding licensing.
- If using the check stock and/or PAD detection functionality (Pre-Authorized Draft and Payee/Payor blacklist detection) a Fraud Protect installation is required. Contact your Kofax business representative for further details.
- If encrypted passwords are used then the Account Loader, Image Loader, Getter, Putter, Day's Final Processing Service Programs as well as the Password Encoder require an updated FraudOne licensing key. Consult your Kofax business representative for further information regarding licensing.

Overview FraudOne Service Programs

The FraudOne Service Programs are standard interfaces that can be used for data processing. The Service Programs may be used out of the box (standard functionality), or customized for the end user's business needs.

The FraudOne Service Programs enable the input and output of signature images, check stock, images and information, and SignInfo information directly in and out of the FraudOne database. This enables the exchange of data within the customer environment.

The first part of this chapter describes the Service Programs, their usage, and their limitations. The second part describes configuration possibilities of the Service Programs.

The following Service Programs are available:

- 1. Account Loader (AL)**
Loads the initial customer, accounts, signatory information into the SignBase database.
- 2. Image Loader (IL)**
Populates the SignBase database with check image references that are used in check stock verification.
- 3. Signature Reference Filter (SRF)**
Used for automatically loading signatures in a batch mode and comparing the similarity of new signatures with variants currently stored in the SignBase database.
- 4. Getter**
Loads data and images captured from payment forms and documents into the SignCheck database for processing.
- 5. Putter**
Creates a file that contains the final results of the SignCheck process.
- 6. Day's Final Processing (DFP)**
A process that empties the daily SignCheck tables and files and prepares it for the following processing cycle.
- 7. Fraud Feedback File Loader (F3 Loader)**
Loads the signatures and check stock images that were processed and filtered by the customer's Fraud Feedback System.
- 8. XML Loader**
Loads data in XML format into the SignBase database.

The following diagram illustrates how each of the individual service programs is used within a typical FraudOne installation:

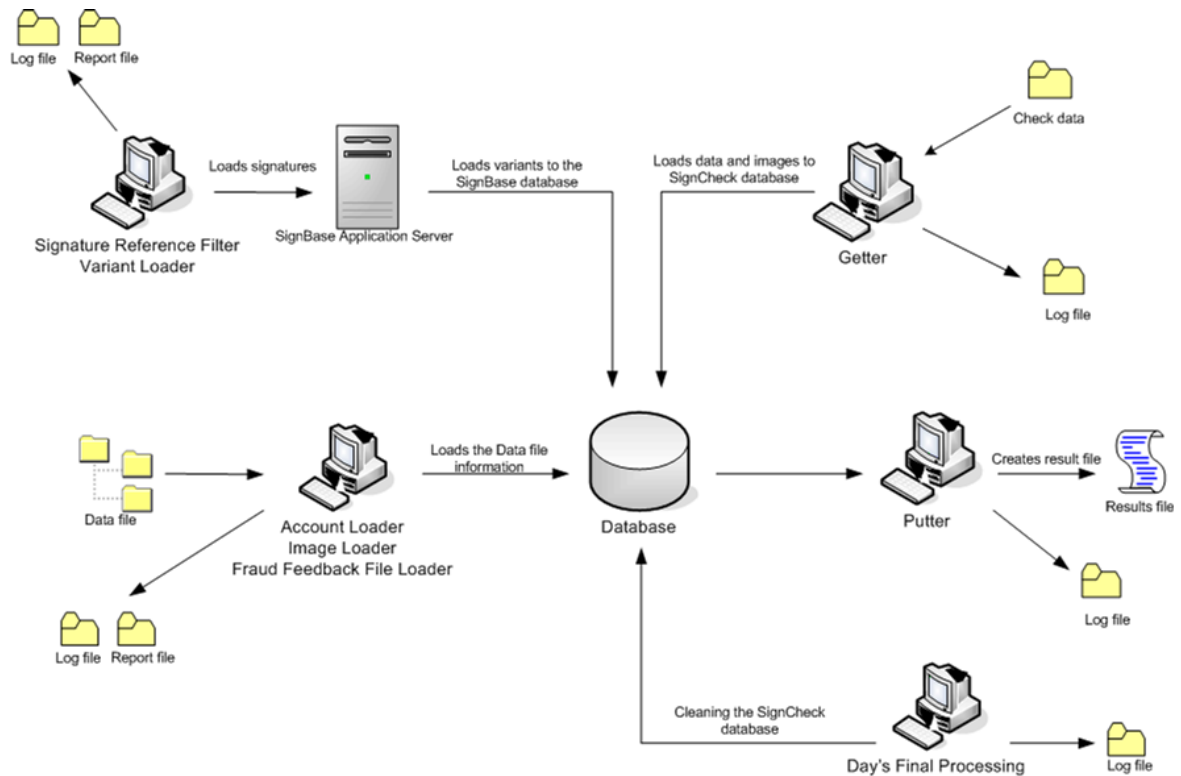


Figure 21: FraudOne Service Programs workflow

Service Programs use and functionality

All of the service programs, except for the Putter and DFP, accept Data files containing the input data to be processed by the program. The Data file is an input file to be processed by a particular service program. An Activate file, used optionally only by the Account Loader, is created in the Data file directory once the Data file is complete and ready to be loaded. In all other cases it is the deliverer's responsibility to ensure that data files are complete as soon as they exist. This can be achieved by the usage of a temporary file name extension (e.g. 'Name.\$\$\$'). When all data are complete, the deliverer renames this temporary file to its final name. When a Data file refers to other files, it is also the deliverer's responsibility to ensure that they exist as soon as the Data file gets its final name.

Therefore, the administrator is required to specify:

- The Data file to be used
- The directory location of the Data file to be used
- The Activate file (Account Loader only)

When started, the Service Programs are customized to retrieve new Data files for processing. Once the processing cycle is complete the Service Programs searches the directory for more Data files at preset intervals.

Standard Service Programs GUI

The Service Programs GUI is a visual interface that allows the user to perform the following:

- Start and Stop the Service Program process
- Locate the Data file to be inputted
- Change the directory of the Data file
- View, filter, and save generated logs
- Change the directory of the Log file
- View the progress of the file processing
- Restarting a process

The Service Programs GUI is invoked by double-clicking the command file. The name and location of the command file is described in each of the Service Program sections.

The standard Account Loader GUI is displayed below:

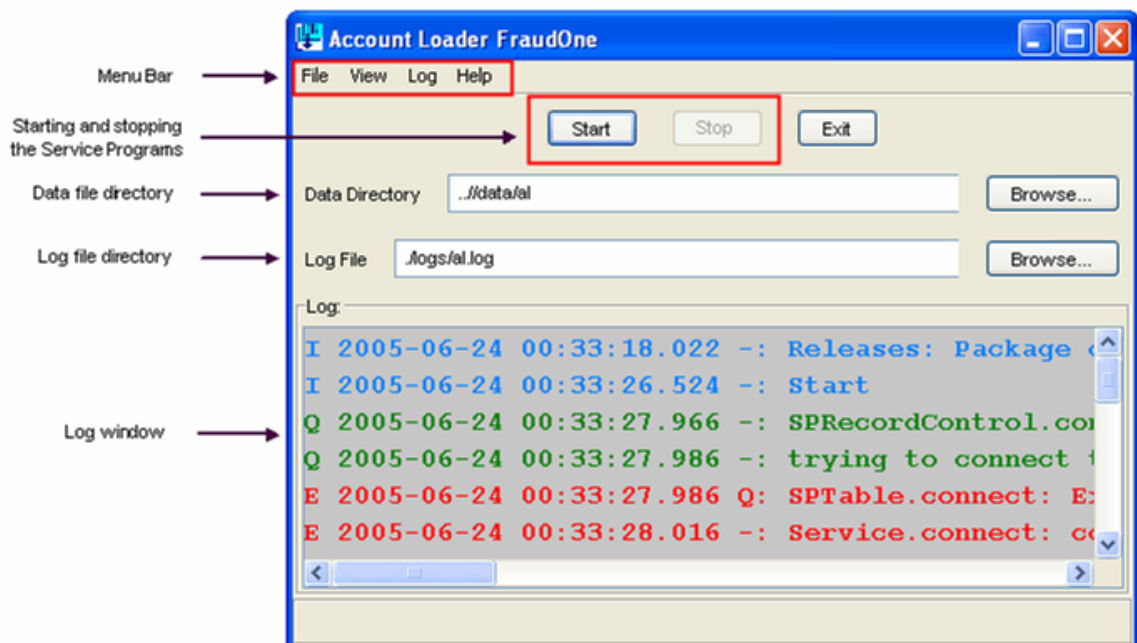


Figure 22: Service Program GUI description

The Menu bar selections allow for the following:

- **File:** save the log to a file on exit
- **View:** view the logs or the file progress
- **Log:** filter the log entries

Log Files

During data processing, the service programs may be configured to output logging information in the form of a log file. If enabled, these logs will record specific events and actions taken at specific times by the service programs during data processing. The Service programs GUI may be used to enable logging for a particular service program.

The Log menu provides the user with the ability to customize the display of the log entries. The Log level can be filtered via the Service Program GUI by clicking **Log** on the menu bar and selecting one or more of the **Trace levels** as shown below:

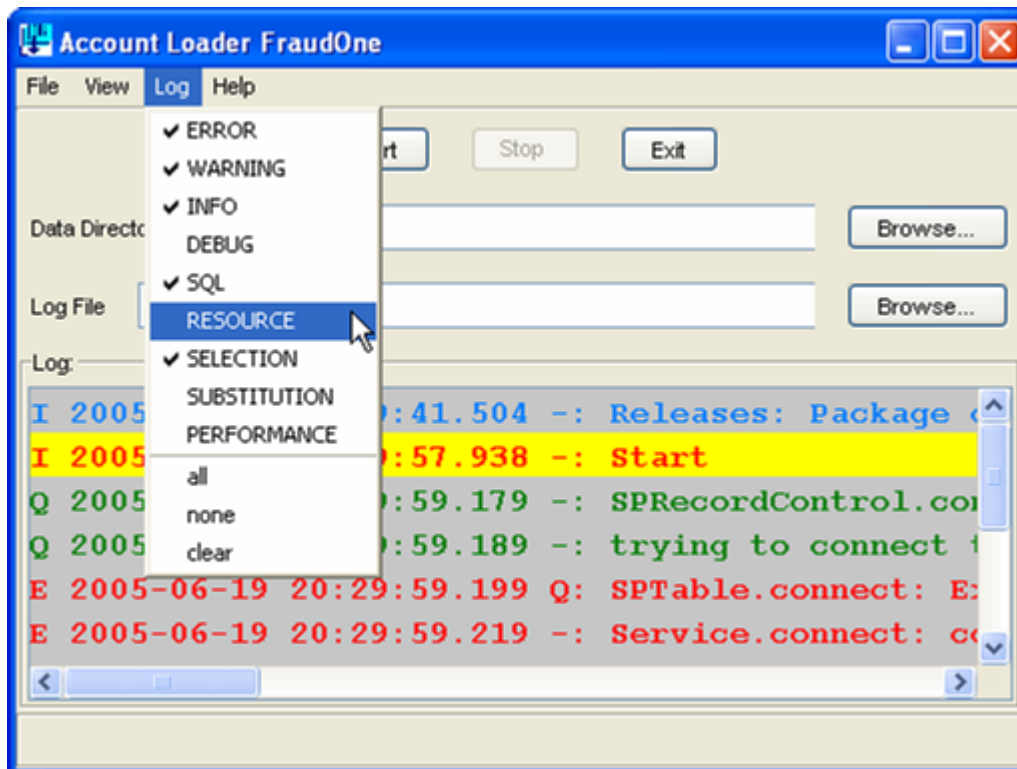


Figure 23: Using Log trace levels

The following describes the available Trace levels for use with the Log file:

- **Error:** all errors including critical ones are written to the Log file
- **Warning:** displays only the warning messages
- **Info:** displays general processing information without error messages
- **Debug:** displays the debugging information
- **SQL:** displays SQL related debugging information
- **Resource:** displays the content of the resource files
- **Selection:** displays a Service Program's selection result and the reason for that particular selection
- **Substitution:** displays a Service Program's substitution process
- **Performance:** displays performance statistics for each run of the Service Program

In order to save the log, select:

File > Save log

The Log file is saved in the Log file directory.

The Log files and Trace levels can also be configured with the **service.properties** file.

Viewing the progress of files

The Batch Progress feature allows the administrator to view the progress for the files currently being processed by the service programs.

In order to use the Batch Progress feature from the Menu bar click **View** and select **Batch Progress**.

In the 'File Type' box select one or more of the following file type to be displayed:

Ready: files that are ready for processing

In progress: files that are currently being processed by the Service Program

Finished: files that have finished the processing cycle

Rejected: files that are rejected for specific reasons

The following displays the Account Loader dialog box with the Batch Progress feature:

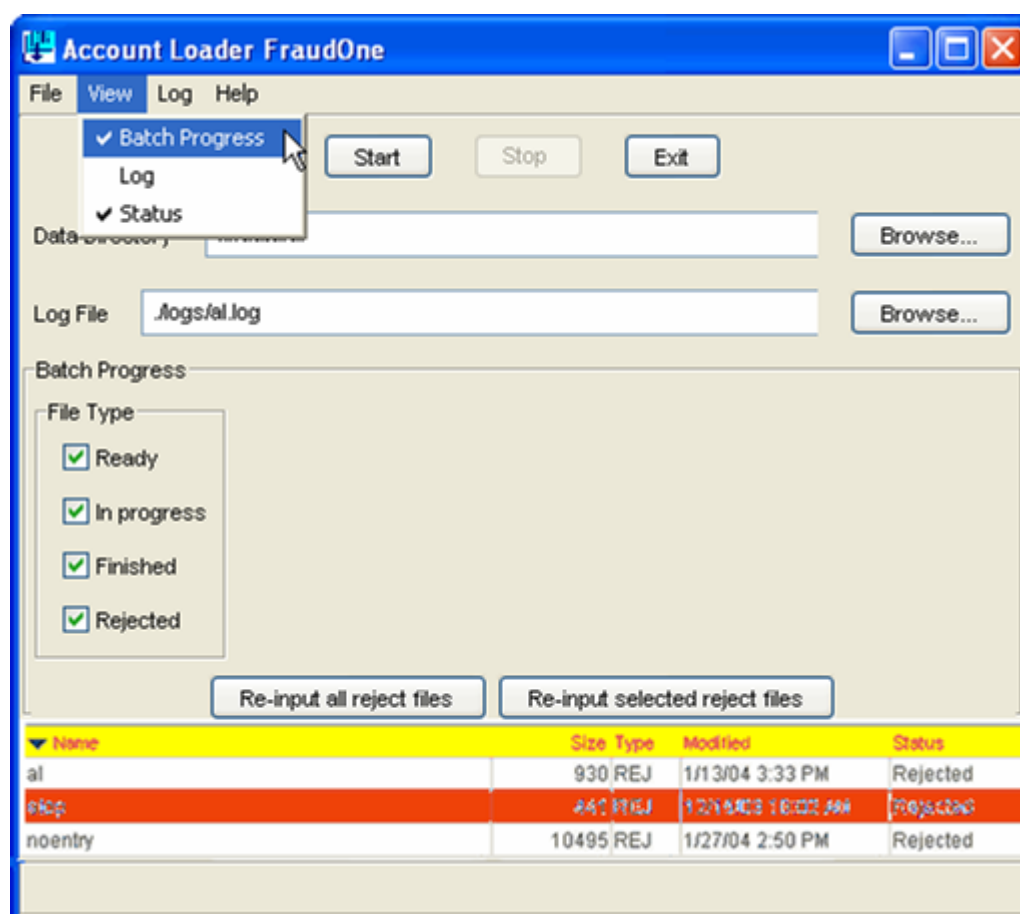


Figure 24: Using Batch Progress

Re-starting a process

If a re-starting of a particular process is required, this may be done via the batch progress feature in the Service Program GUI described in the previous section.

In order to reprocess a rejected file click on the rejected file(s) that needs to be re-processed. The rejected file line will be highlighted in red. Click the **Re-input selected reject files** button. Optionally, all the reject files can be reloaded by clicking **Re-input all reject files** button.

Using the Data Files

The Data file is an input file to be processed by a particular service program. The file may contain data describing a customer, account, signatory information, check stock images, and/or actions to be taken. The Data file is in ASCII format and is 'line-based'. Each line is unique and contains data specific to a particular account or customer that a given Service Program is to process. Service program Data files can be distinguished by the fact that they are assigned a particular suffix. The suffixes used for a particular data file is configurable via the **service.properties** configuration file located in the Central Configuration database.

In order to input only "new" Data files from the SignPlus directory the Service Program uses the following naming principles:

- Inputted Data files are in the name format **name.dat**
- As soon as the Service Program processes a Data file it outputs a file **name.wrkr**. This **'wrkr'** work file includes the point of the last line that was processed; this avoids reprocessing if the Service Program is interrupted and then restarted.
- Any rejected data lines will be added to a reject file **name.rej**. A line may be rejected if the information does not exist or does not correspond to the SignPlus database. The Getter Service Program does not use reject files.
- Erroneous Data files are renamed **name.err**. This applies to a Data file containing information that cannot be processed. The Getter Service Program does not use this naming convention.
- As soon as the Service Program finishes processing a file the **name.wrkr** file will be deleted. The "name.dat" file can then be renamed, moved or deleted.

When invoked, each service program searches in the pre-defined directory for Data files with a matching suffix. All other files in the directory whose suffix does not match the pre-defined one will be ignored. When a service program has finished processing its data file, it will rename it by changing its suffix. The renamed file will be stored in the same directory as the original input data file.

This section detailed the standard use and naming conventions of the Data file. Optional parameters, including the suffix to search for, can be configured with the **service.properties** file. The use and functionality of the 'work' and 'reject' files are further described in the 'Service Programs configuration' section below.

Important Due to their different functionalities, the Putter and Day's Final Processing Service Programs do not use Data files. Refer to Putter and Day's Final Processing for more details on their functionalities.

Using the Data Viewer

The FraudOne Data Viewer tool enables viewing Data files of all the Service Program except the F3-Loader and the Day's Final Processing.

The Data Viewer GUI enables the display of all lines of the Data file. By double-clicking a specific line, the column's values of all tables of this line are shown. An image can be displayed by double-clicking on a table-row which contains an image. The image will be shown on the right side of the screen. This window can display two images. If this image contains a signature snippet, the border of this snippet is shown as a green frame.

The **service.properties** file enables for the configuration of the Data Viewer.

The Data Viewer GUI is started by the command file located in the FraudOne installation directory. Every Service Program has a different extension to the DV command file name. **DVG.cmd** is Getter specific and **DVIL.cmd** is Image Loader specific.

In order to open a file, select

File > Open Data file

or drag and drop the Data file into the Data Viewer GUI.

Using Report Files with the Service Programs

Report files record the actions and database changes made by the Service Programs (except Day's Final Processing, Getter and Putter). The service programs output these files during the data processing to record the actions that have been taken. The reports may consist of one or more

report data lines sorted in alphabetical order. The 'Report file' section below details the use and configuration of report files with the FraudOne Service Programs.

Service Programs configuration

The **service.properties** file, located in the Central Configuration database, is used for the configuration of the Service Programs. It allows for the configuration of the following standard settings:

- Determine Trace levels for the logs
- Log file naming and handling
- Directory of the Data file
- Naming and of the Data file

Other available configuration settings depend on the Service Program used. [The service.properties configuration file](#) section describes other possible configurations of the Service Programs using this property file.

Service Programs

Account Loader

The FraudOne Account Loader (AL) is used to automatically insert, update or delete the customer, account and signatory information in the SignBase database. For this purpose the Account Loader uses a standardized file interface directly connected to the database. The Account Loader is provided to enable the 'initial load' of information into the SignBase database, and to perform the daily data updates.

The Account Loader processes the customer, account and/or signatory data that is loaded from the customer information system and writes the information to the database.

The Account Loader can be used to:

- Create, modify, and delete customers (including historization of current entries)
- Create, modify, and delete accounts (including historization of current entries)
- Create signatories
- Create signature variants
- Create simple rules
- Create mono and gray signatures (including cleaning)

Account Loader process

The Account Loader loads the Data file that is specified by the administrator. The Data file contains data that is extracted and converted from the bank's Customer Information System database (CIS). The file is an ASCII 'line-based' file where each line contains unique data describing the account, customer, signatory information, and action to be taken. If a signature variant is to be loaded, the data line also contains the full path to the image.

When started, the Account Loader reads the file 'line by line' and imports the data into the SignBase database. An Activate file, used only by the Account Loader, is a trigger that signals the Account Loader that the Data file is complete and ready to be loaded. Since the Account Loader searches for a Data file at predefined times it is important that an Activate file is used so Data files are not loaded while being copied or moved that directory. This ensures that only a Data file with complete information is loaded by the Account Loader.

The Account Loader searches for an Activate file and then its corresponding Data file. Depending on the initial customization, the Account Loader can search for an Activate file first and then the corresponding Data file or to search only for the Data file. The activate file usually contains the same name as the data file; the Data file uses the **'.dat'** file extension and the Activate file uses **'.cnt'** file extension.

After the successful processing of a data file, this data file and its corresponding Activate file can be moved, renamed, or deleted.

The action taken after a Data file is processed is described below:

- **Move:** After processing, the Data file retains the same name and suffix but is moved to another directory. This is configured with the **'renamePath'** key.
- **Rename:** After processing, the Data file can be renamed and saved in the original directory. This is configured with the **'renameSuffix'** key.
- **Delete:** By default, the Account Loader deletes the Data file and Activate file after processing.

If the processing of a Data file fails, its suffix is renamed to **'.err'** in order to avoid a second processing by the Account Loader.

The action taken on a processed Data file is configured with the **service.properties** file and further described in the [Service Programs configuration](#) section.

The Account Loader technical description can be found in the document *FraudOne Service Programs Interfaces*, which further describes the standard Data file format. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Installing the Account Loader

See the *Kofax FraudOne Installation and Migration Guide* regarding proper installation procedures.

Starting the Account Loader

There are three ways to start the Account Loader:

1. The Account Loader is started by using the command file **AccountLoader.cmd** located in the FraudOne installation directory. To start the Account Loader simply double-click the command file. This initiates the Account Loader GUI. In order to start the Account Loader process, simply click the **Start** button.

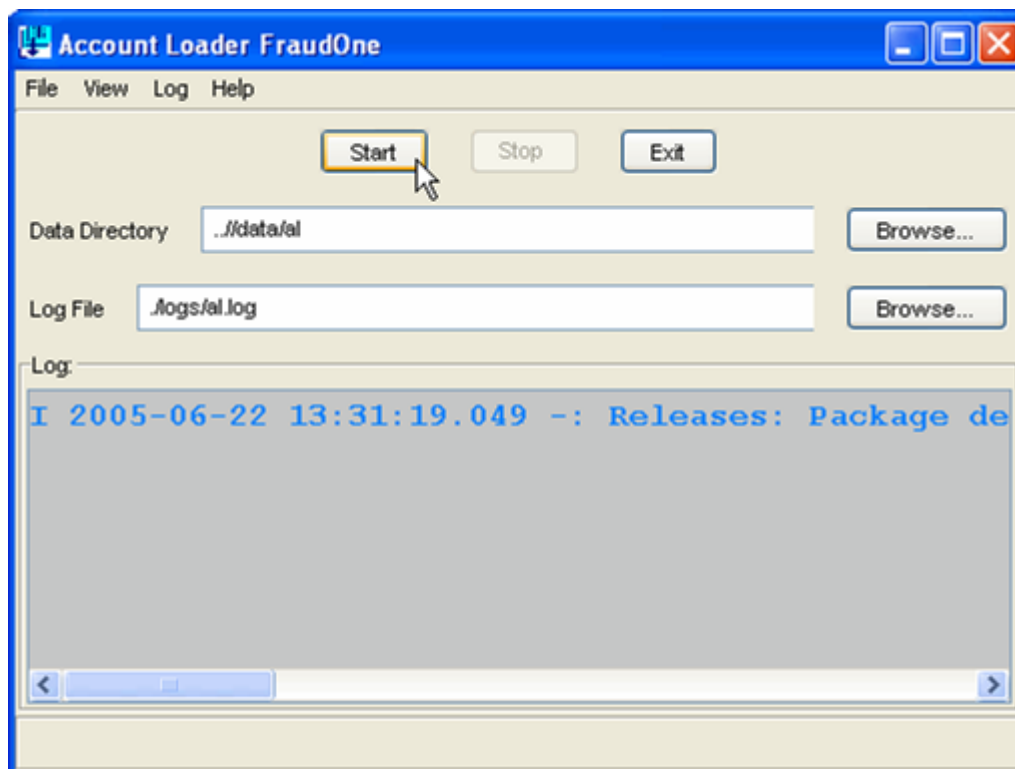


Figure 25: Account Loader GUI

2. If specified in the **SrvMngr4.ini** file, the AL can be initiated automatically when Server Manager starts or it can be scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the AL is loaded by the Server Manager but does not start automatically, you may also use the Server Monitor. Locate the AL on the left window of the Server Monitor dialog box, right-click on **Account Loader**, and then select **Start Server**. The same functionality provided by the Account Loader GUI functionality is also available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the Account Loader

The performance of the machine where the AL is started may not be able to handle more than one instance since each AL instance requires an instance of the Java Virtual Machine. Depending on the current environment configuration and the processing load, the Account Loader may experience decreased performance.

Image Loader

The FraudOne Image Loader (IL) is provided for enabling the population of the SignBase database with references that are to be used for check stock verification.

The Image Loader is only needed if using the Check Stock Verification application functionality. The Image Loader typically will be provided with check stock images to load from the following sources:

- The 'inclearing' items processing

- An image archive

It is the administrator's responsibility to provide these images by adhering to the IL input requirements. The IL loads the check images into the SignBase database according to pre-defined rules (e.g. maximum number of image references allowed for an account), as well as specific quality criteria.

The Image Loader verifies check images for the following quality criteria:

- If the check amount is within specified range
- If the check stock image is within predefined size range
- If the check image is an Image Replacement Document
- If the check image is a Pre-Authorized Draft
- If the check image fulfills the minimum serial number requirement
- If the similarity of the check images falls above a defined match rate

The check image selection process goes through several evaluation and validation steps before a check image is added to the reference database.

Image Loader process

The Image Loader loads the Data file that is specified by the administrator. The Data file contains data that is extracted and converted from the bank's Customer Information System database (CIS). The file is an ASCII 'line-based' file where each line contains unique data describing the check information and quality criteria. The data line also contains the full path of the image to be loaded. The Image Loader reads the file 'line by line' and imports the data into the SignBase database.

After the successful processing of a data file it can be moved, renamed, or deleted.

The action taken after a Data file is processed is described below:

- **Move:** After processing, the Data file retains the same name and *suffix* but is moved to another directory. This is configured with the '**renamePath**' key.
- **Rename:** After processing, the Data file can be renamed and saved in the original directory. This is configured with the '**renameSuffix**' key.
- **Delete:** By default, the Image Loader deletes the Data file after processing.

If the processing of a Data file fails, its suffix is renamed to '**.err**' in order to avoid a second processing by the Image Loader.

The action taken on a processed Data file is configured with the service.properties file and further described in the [Service Programs configuration](#) section.

The standard input format is described in the document *FraudOne Service Programs Interfaces*. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Installing the Image Loader

See the *Kofax FraudOne Installation and Migration Guide* regarding proper Image Loader installation procedures.

Starting the Image Loader

There are three ways to start the Image Loader:

1. The Image Loader is started by using the command file **ImageLoader.cmd** located in the FraudOne installation directory. To start the Image Loader simply double-click the command file. This initiates the Image Loader GUI. In order to start the Image Loader process, simply click the **Start** button.

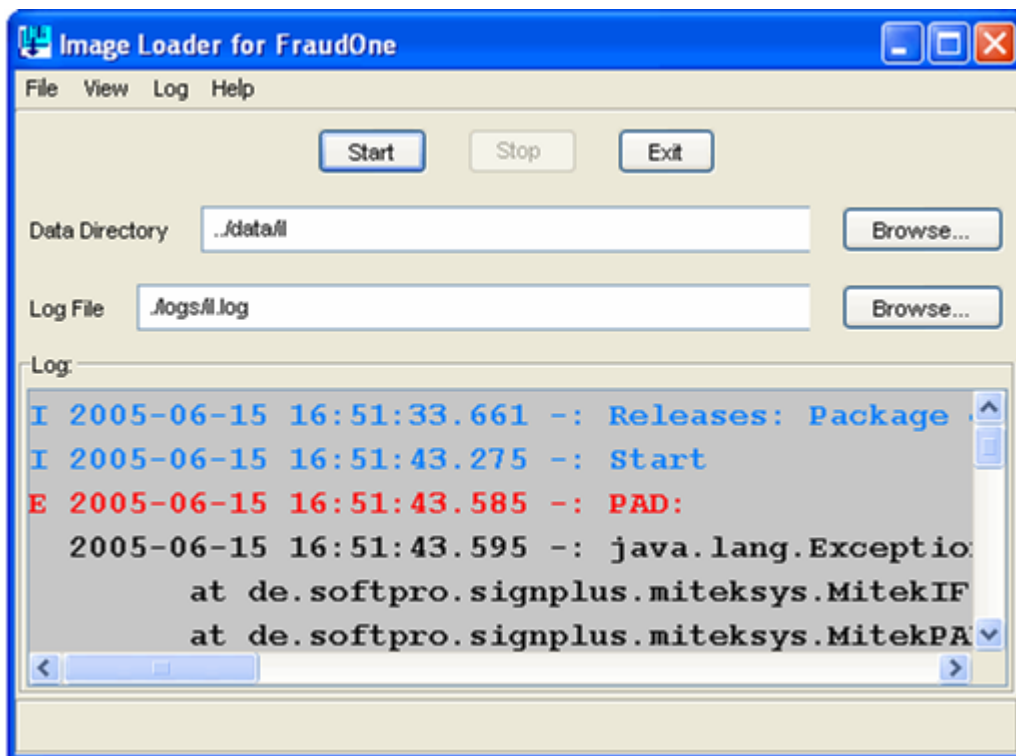


Figure 26: Image Loader GUI

2. If specified in **SrvMgr4.ini** file, the IL can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMgr4.ini** file.
3. If the IL is loaded by the Server Manager but does not start automatically, you may use the Server Monitor. Locate the Image Loader on the left window of the Server Monitor dialog box, right-click the **Image Loader** and then select **Start server**. The Image Loader GUI functionality is available when using the Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the Image Loader

The performance of the machine where the IL is started may not be able to handle more than one instance since each IL instance requires an instance of Java Virtual Machine. Depending on the current environment configuration the Image Loader may experience decreased performance.

Signature Reference Filter

The FraudOne Signature Reference Filter (SRF) is an optional program used to speed up the capturing of new signatures to be used with Automatic Signature Verification. It uses signature comparison techniques in order to optimize the variant signature loading process by reducing the amounts of 'similar' signatures.

The SRF processes signature information that is loaded from the Data file. When loading a new signature into the SignBase database, the SRF compares the new signature derived from a paper-based payment transaction with all existing signatures currently stored for a specific account or customer on the database. If a similar signature is not found among the current references, it will store the new signature as a new variant for that corresponding account or customer. If the signature is found to be too similar to an existing one, it will not be stored as a new variant signature. This prevents the database from storing unnecessary data and improves the data transfer rate of the SignCheck Automatic Signature Verification (ASV) process by reducing the number of comparisons necessary during the verification process.

'Similar' signatures are defined as two signatures whose resulting match rate is higher than the 'referenceMatchRate' settings which are defined in the **service.properties** file.

SRF supports both 'on-line' and 'off-line' modes:

- In the on-line mode, the SRF is connected to the server and communicates directly with the SignBase database.
- In the off-line mode, the SRF performs a comparison of the signatures from a loaded input file containing signature variants.

Before a variant can be used it must be assigned to existing authorized signatories. Using assigned variants with reference signatures in the SignBase database can reduce the time needed when introducing Automatic Signature Verification in payment transactions.

The SRF can also enter the variants in a way that they are automatically bound to and stored with a "dummy" signatory. A dummy signatory has no data (first name and family name) and includes full signing rules.

When loading signatures, the SRF can detect the correct signature area and crop the signature snippet for the verification process using the '**crop.**' prefix specified in the **service.properties** file described in the [Service Programs Configuration](#) section.

Signature Reference Filter process

The SRF loads the Data file that is specified by the administrator. The Data file contains data that is extracted and converted from the bank's Customer Information System database (CIS). The file is an ASCII 'line-based' file where each line contains unique data describing the signature information.

The data line also contains the full path of the check signatures to be loaded. When started, the SRF reads the file 'line by line', and imports the data into the SignBase database.

After the successful processing of a data file it can be moved or renamed.

The action taken after a Data file is processed is described below:

- Move: After processing, the Data file retains the same name and suffix but is moved to another directory. This is configured with the '**enamePath**' key.
- Rename: After processing, the Data file can be renamed and saved in the original directory. This is configured with the '**renameSuffix**' key.

If the processing of a Data file fails, its suffix is renamed to '**.err**' in order to avoid a second processing by the SRF.

The action taken on a processed Data file is configured with the **service.properties** file and further described in the [Service Programs Configuration](#) section.

The SRF technical description can be found in the document *FraudOne Service Programs Interfaces*, which describes the file interface between SignCheck and the payment transactions system. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

The signature comparison function works in conjunction with the SIVAL neural-network engine, requiring licensing of the module. Consult your Kofax business representative for further information regarding licensing.

Installing the Signature Reference Filter

See the *Kofax FraudOne Installation and Migration Guide* regarding proper Signature Reference Filter installation procedures.

Starting the Signature Reference Filter

There are three ways to start the Signature Reference Filter:

1. The Signature reference Filter is started by using the command file **SRF.cmd** located in the FraudOne installation directory. To start the SRF simply double-click the command file. This initiates the SRF GUI. In order to start the SRF process, simply click the **Start** button.

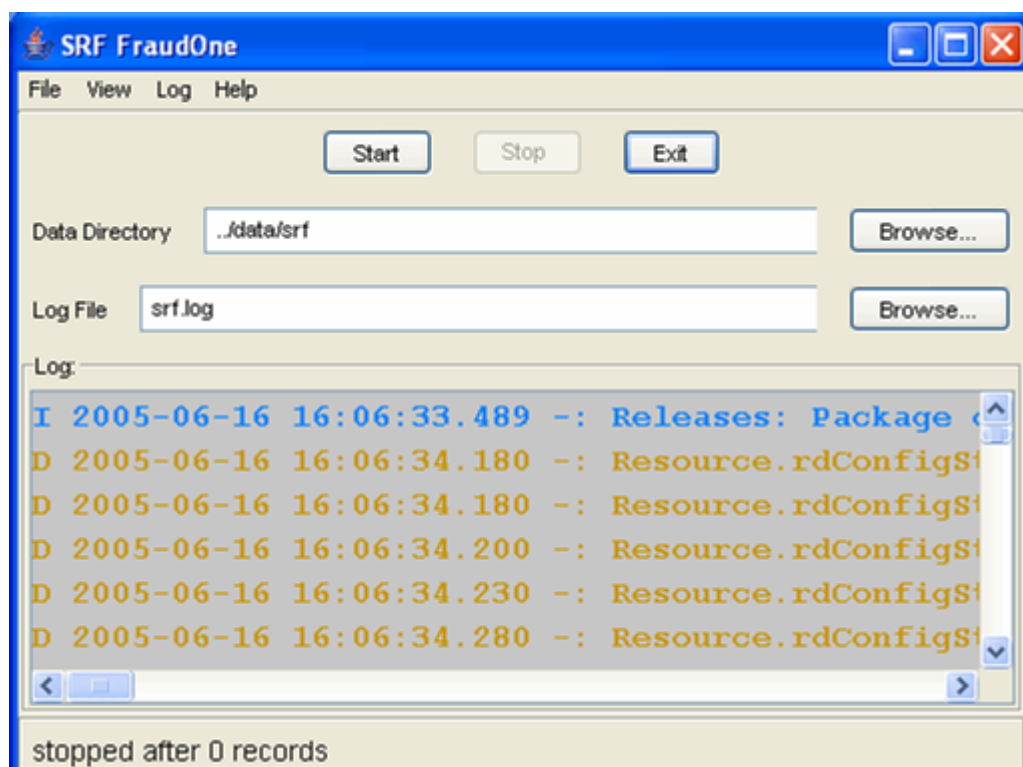


Figure 27: Signature Reference Filter GUI

2. If specified in **SrvMngr4.ini** file, the SRF can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the SRF is loaded by the Server Manager but does not start automatically, you may use the Server Monitor. Locate the SRF on the left window of the Server Monitor dialog box, right-click the **Service Program** and then select **Start server**. The SRF GUI functionality is available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the Signature Reference Filter

Depending on the current environment configuration the Signature Reference Filter may experience decreased performance for the following reasons:

Since SRF is a SignBase client there cannot be more than two instances due to the limitation of the Server Manager.

The performance of the machine where the SRF is started may not be able to handle more than one instance since each SRF instance requires an instance of Java Virtual Machine.

SignCheck Getter

The SignCheck Getter is a standard interface that loads data and images, captured from payment forms and documents, into the SignCheck database for processing. This is the program you will need to use for loading inclearing items to be processed by SignCheck. The input data for the Getter consists of the signature image and transaction information - amount to be paid, account information, and clearing date. Prior to loading, the Getter verifies check images for the following characteristics:

- Special image compression
- Pre-Authorized Drafts (PAD) (if enabled)
- Image Replacement Documents (IRD)
- Size difference indicating microline correction slips, check jackets, and irregular sizes

The Getter can be configured to load only specific transaction forms (inclearing items) into the system. This can be configured based on the following criteria:

- Amount limit
- Account numbers
- Partial quantity of all documents
- Transaction code

For example, it is possible to load only those items whose amount is below a specified threshold, or that belong to a particular transaction type.

The Getter fills the SignCheck database with document data from payment transactions. The data is provided as an input file by the administrator. The data can include entries from a specific time period, from a specific event, or by direct reading from an external source. The Getter technical description can be found in the document *FraudOne Service Programs Interfaces*, which describes the file interface between SignCheck and the payment transactions system.

The Getter also provides a Pre-Authorized Draft (PAD) detection mechanism which requires the use of the Fraud Protect engine. If PAD Detection is enabled the Getter performs a verification of

the input images for PAD characteristics. Depending on the type of setup, the Getter can also perform additional verification steps before inputting the PAD to the FraudOne workflow.

Following are the optional levels of PAD configurations, specified in the **service.properties** file:

1. PAD detection match (range 0-1000)
2. Account holder/payor match (range 0-1000)
3. Blacklisted account/payee match (range 0-1000)

This process is done sequentially and the PAD is loaded with the match rate.

Depending on the configuration, the Getter can also load Image Replacement Documents (IRD) with a unique result code (flag) into the workflow for processing. This item is loaded into the Getter file as an IRD.

The Getter is also used for cropping signature snippets from the incoming data. This may be done statically by specifying a crop area that is specified as a relative position from the bottom right corner of the incoming image. Alternatively, the getter may be configured to automatically detect the signature within the incoming item.

These features are enabled using 'crop.' prefix in the **service.properties** file, described in the [Service Programs configuration](#) section.

SignCheck Getter process

The Getter loads the Data file that is specified by the administrator. The Data file contains data that is extracted and converted from the bank's Customer Information System database (CIS). The file is an ASCII 'line-based' file where each line contains unique data describing the information to be loaded as well as actions to be taken in the workflow. The data line also contains the full path of the image to be loaded. When started, the Getter reads the file 'line by line', and imports the data into the SignBase database.

After the successful processing of a data file it can be moved or renamed.

The action taken after a Data file is processed is described below:

- **Move:** After processing, the Data file retains the same name and suffix but is moved to another directory. This is configured with the '**renamePath**' key.
- **Rename:** After processing, the Data file can be renamed and saved in the original directory. This is configured with the '**renameSuffix**' key.
- By default, the Getter renames the Data file after processing to a file with the same name but with '**da_**' suffix

The action taken on a processed Data file is configured with the **service.properties** file and further described in the [Service Programs Configuration](#) section.

The standard input format is described in the document *FraudOne Service Programs Interfaces*. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Installing the SC Getter

See the *Kofax FraudOne Installation and Migration Guide* document regarding proper Getter installation procedures.

Starting the SC Getter

There are three ways to start the Getter:

1. The Getter is started by using the command file **Getter.cmd** located in the FraudOne installation directory. To start the Getter simply double-click the command file. This initiates the Getter GUI. In order to start the Getter process, simply click the **Start** button.

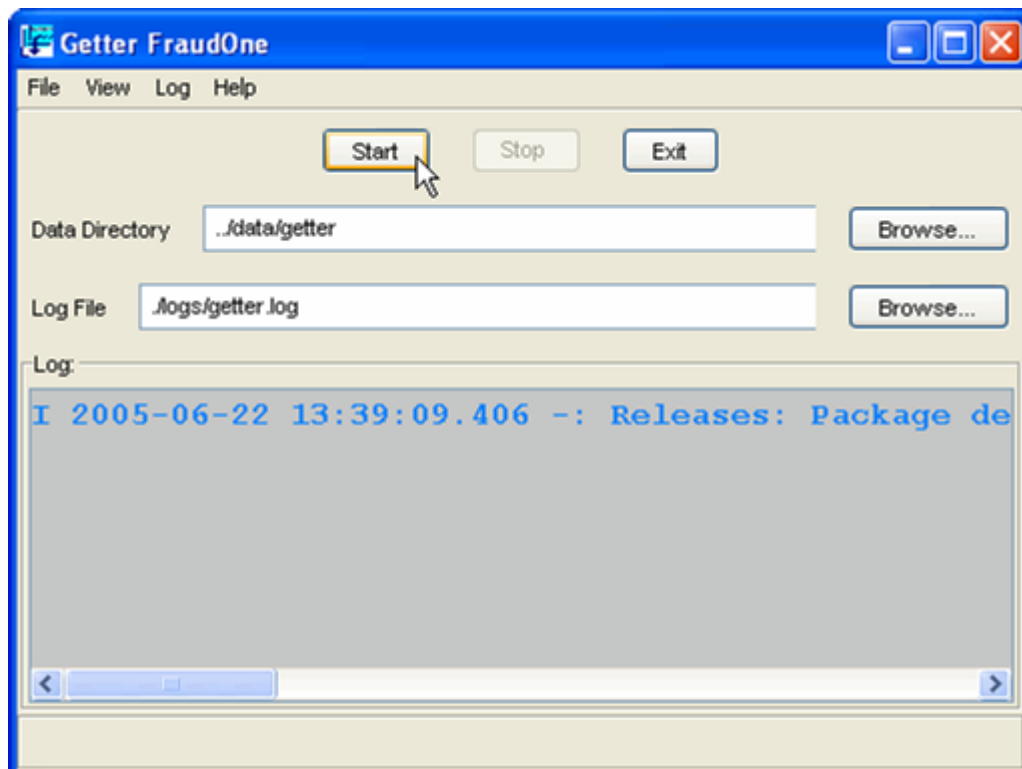


Figure 28: Starting the Getter process

2. If specified in **SrvMngr4.ini** file, the Getter can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the Getter is loaded by the Server Manager but does not start automatically, you may use the Server Monitor. Locate the Getter on the left window of the Server Monitor dialog box, right-click the **Service Program** and then select **Start server**. The Getter GUI functionality is available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the SC Getter

The performance of the machine where the Getter is started may not be able to handle more than one instance since each Getter instance requires an instance of Java Virtual Machine.

Depending on the current environment configuration the Getter may experience decreased performance.

SignCheck Putter

SignCheck provides a standard file interface for the final results of the SignCheck process. This final output file is produced by the Putter Service program. To create the output file, the Putter Service Program reads the results from the SignCheck workflow and creates a single result line for each of the items that was input by the Getter for processing by SignCheck. The result file contains the transaction information of each item and the results of each of the verification steps that the item was processed within the SignCheck workflow. The output file is referred to as the "Putter File".

This enables the evaluation of results files in order to:

- Store results in an archive and audit system
- Merge the data with the results of debiting systems
- Filter out rejected items for review by fraud departments

After the results are output with the Putter program, the document status in the SignCheck workflow is set to a value that allows no further processing. i.e. Workflow processing ends as soon as the Putter has output its results for a particular item.

Physical storage of the Putter file can be either on the same file server that SignCheck receives the document files or stored on another location. The Putter files are generally used for archiving, booking, and for sorting out the return items in check-based installations.

The Putter technical description can be found in the document *FraudOne Service Programs Interfaces*, which describes the file interface between SignCheck and the payment transactions system.

Important Multiple Putter programs may run at the same time serving multiple BNOs. However, one Putter must serve one BNO.

Use of Primanota processing

The name of the result file depends on whether Primanota processing is enabled. Primanota is a batch of checks where all the checks belonging to a specific Primanota (i.e. group) are contained within one file. All the lines in the Data file of a Primanota contain the same value number associating it to that Primanota. When using the Primanota functionality, the Putter writes the result file for a Primanota only when all checks for the Primanota are fully processed.

Installing the SC Putter

See the *Kofax FraudOne Installation and Migration Guide* regarding proper Putter installation procedures.

Starting the SC Putter

There are three ways to start the Putter:

1. The SignCheck Putter is started by using the command file **Putter.cmd** located in the FraudOne installation directory. To start the Putter simply double-click the command file. This initiates the Putter GUI. In order to start the Putter process, simply click the **Start** button.

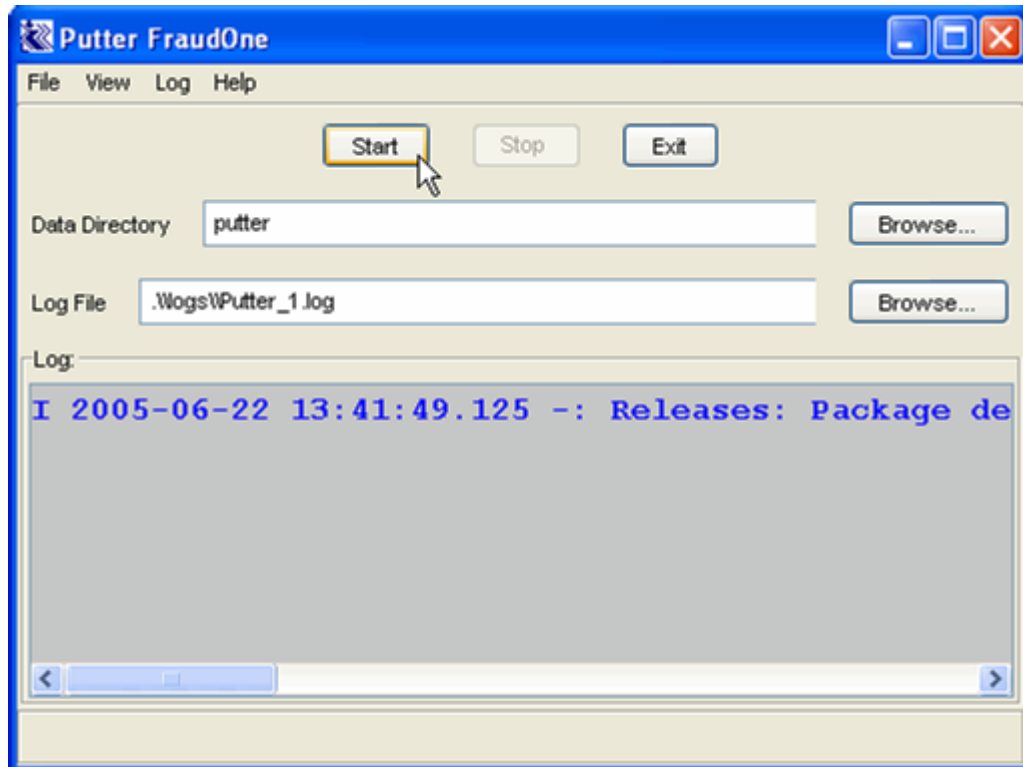


Figure 29: Starting the Putter process

2. If specified in **SrvMngr4.ini** file, the Putter can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the Putter is loaded by the Server Manager but does not start automatically, you may use the Server Monitor: Locate the Putter on the left window of the Server Monitor dialog box, right-click the **Service Program** and then select **Start server**. The Putter GUI functionality is available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the SC Putter

The default configuration of the Putter allows only for one instance to be used at one time. Having more than one Putter instance must be initially configured by Kofax. Consult your Kofax business representative for further details.

Day's Final Processing

The Day's Final Processing (DFP) is a process usually performed at the end of each working day. This final component of the processing cycle empties the SignCheck tables and files from data used in the processing of the current day items. This prepares SignCheck database and the processes for a new item processing cycle.

The DFP technical description can be found in the document *FraudOne Service Programs Interfaces*, which describes the file interface between SignCheck and the payment transactions system.

Installing the Day's Final Processing

See the *Kofax FraudOne Installation and Migration Guide* regarding proper Day's Final Processing installation procedures.

Starting the Day's Final Processing

There are three ways to start the Day's Final Processing:

1. The DFP is started by using the command file **DFP.cmd** located in the FraudOne installation directory. To start the DFP simply double-click the command file. This initiates the DFP GUI. In order to start the DFP process, simply click the **Start** button.

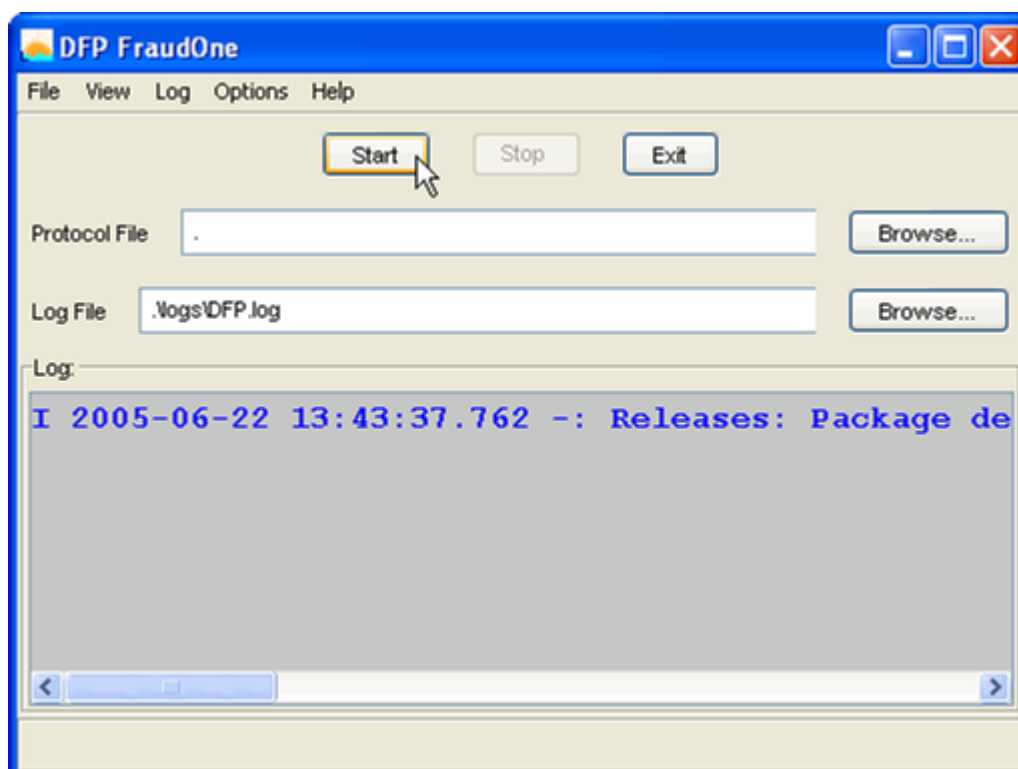


Figure 30: Starting the DFP process

2. If specified in **SrvMngr4.ini** file, the DFP can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the Server Manager chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the DFP is loaded by the Server Manager but does not start automatically, you may use the Server Monitor. Locate the DFP on the left window of the Server Monitor dialog box, right-click the **Service Program** and then select **Start server**. The DFP GUI functionality is available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the Day's Final Processing

The FraudOne functionality allows for only one instance of DFP running at one time.

Fraud Feedback File Loader

The Fraud Feedback File Loader (F3-Loader) is a reference database maintenance program that is used to process the input derived from a bank's Fraud Feedback System. If the bank wishes to provide information about fraudulent items that were found over the course of the processing cycle, these items can be reloaded into the SignBase database and be explicitly marked as being fraudulent. Typically, the fraud feedback information will be loaded back into the database via the F3 loader on a daily basis. This provides the ability to mark reference data as being invalid so that this will be considered during automatic verification.

The following illustration describes the F3-Loader and the bank's FFS workflow:

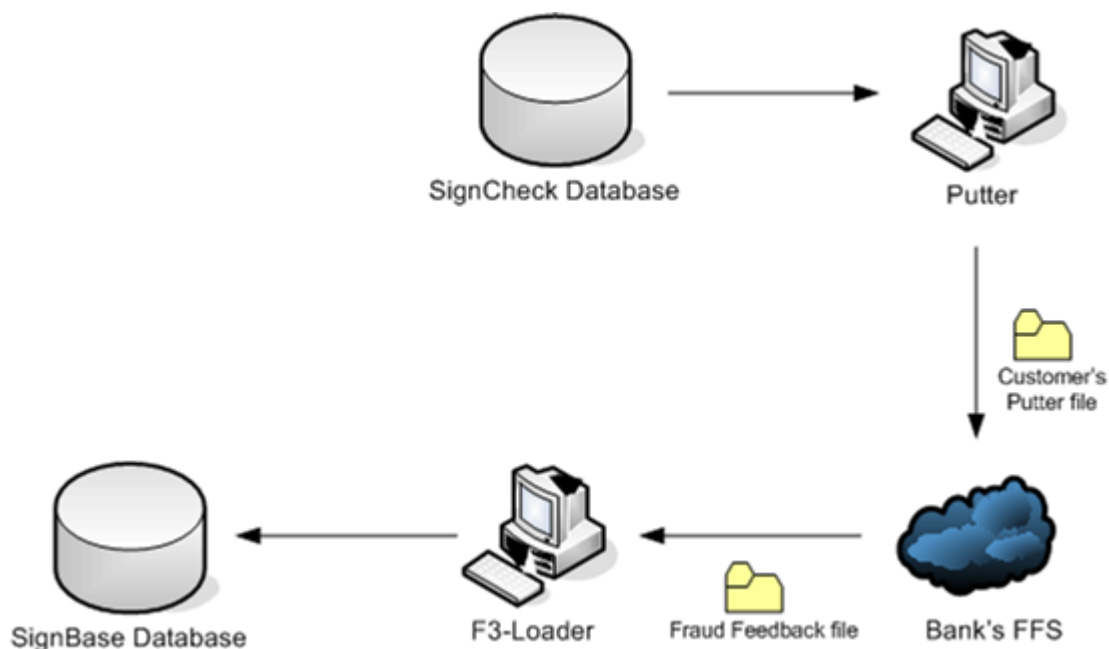


Figure 31: F3-Loader and FFS workflow

F3-Loader Process

The F3-Loader periodically searches the F3-Loader directory for the daily Fraud Feedback File (Data file). This Data file contains information on the check inputs as well as the actions to be taken. The 'action to be taken' by the F3-Loader is as follows:

- The 'D' (delete) action indicates that the corresponding database item should be deleted from the database. This is a logical delete, leaving the records in the 'historized' mode.
- The 'B' (block) action indicates that the corresponding item should remain in the system but prohibited for verification purposes. The following actions takes place:
 - Signatures will be marked as "fraudulent"

- Images will be marked as "not for APIA" i.e. the reference image can no longer be used as a reference image for check stock verification
- Accounts will be marked as "not for ASV" which will also block APIA i.e. the reference image can no longer be used as a reference image for automatic signature verification

After the successful processing of a data file it can be moved, renamed, or deleted.

The action taken after a Data file is processed is described below:

- **Move:** After processing, the Data file retains the same name and suffix but is moved to another directory. This is configured with the '**renamePath**' key.
- **Rename:** After processing, the Data file can be renamed and saved in the original directory. This is configured with the '**renameSuffix**' key.
- **Delete:** By default, the F3-Loader deletes the Data file after processing

If the processing of a Data file fails, its suffix is renamed to '.err' in order to avoid a second processing by the F3-Loader.

The action taken on a processed Data file is configured with the **service.properties** file and further described in the [Service Programs Configuration](#) section.

The standard input format is described in the document *FraudOne Service Programs Interfaces*. Refer to the [Related documentation](#) chapter for access to this and other available Kofax documentation.

Installing the Fraud Feedback File Loader

See the *Kofax FraudOne Installation and Migration Guide* regarding proper F3-Loader installation procedures.

Starting the Fraud Feedback File Loader

There are three ways to start the F3-Loader:

1. The F3-Loader is started by using the command file **F3.cmd** located in the FraudOne installation directory. To start the F3-Loader, simply double-click the command file. This initiates the F3-Loader GUI. In order to start the F3-Loader process, simply click the **Start** button.

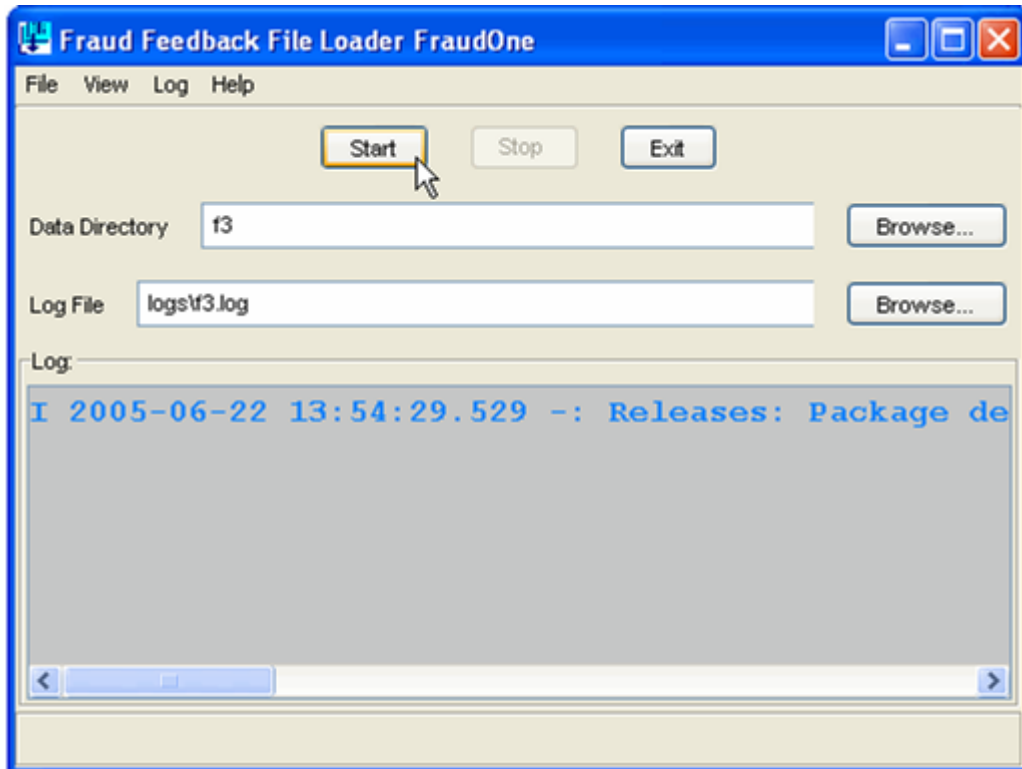


Figure 32: Starting the F3-Loader process

2. If specified in **SrvMngr4.ini** file, the F3-Loader can be initiated automatically when Server Manager starts or scheduled to start and stop at predefined times. Refer to the [FraudOne Server Manager](#) chapter which includes configuration possibilities using the **SrvMngr4.ini** file.
3. If the F3-Loader is loaded by the Server Manager but does not start automatically, you may use the Server Monitor. Locate the F3-Loader on the left window of the Server Monitor dialog box, right-click the Service Program and then select **Start server**. The F3-Loader GUI functionality is available when using the FraudOne Server Monitor. Refer to [Server Monitor](#) chapter for more information.

Limitations of the Fraud Feedback File Loader

The performance of the machine where the F3-Loader is started may not be able to handle more than one instance since each F3-Loader instance requires an instance of Java Virtual Machine.

Depending on the current environment configuration, the F3-Loader may experience decreased performance.

XML-Loader

The XML Loader is a reference database maintenance program that is used load XML-formatted data according the **signplus.dtd** into the Signbase database.

XML-Loader process

The XML-Loader periodically searches the XML-Loader directory for data to be loaded into the SignBase database.

After the successful processing of a data file it can be renamed, or deleted.

The action taken after a data file is processed is described below:

- **Rename:** By default, after processing, the data file can be renamed and saved in the original directory. This is configured with the '**renameSuffix**' key.
- **Delete:** the F3-Loader deletes the Data file after processing

If the processing of a data file fails, its suffix is renamed to ***.err** in order to avoid a second processing by the XML-Loader.

The action taken on a processed data file is configured with the **service.properties** file and further described in the [Service Programs Configuration](#) section.

The input format is described in the **signplus.dtd** file.

Limitations of the XML-Loader

The performance of the machine where the XML-Loader is started may not be able to handle more than one instance since each XML-Loader instance requires an instance of Java Virtual Machine.

Depending on the current environment configuration, the XML-Loader may experience decreased performance.

Configuration of the Service Programs

This section describes the configurations of the FraudOne Service Programs that can be performed by an administrator.

Important Consult your Kofax support representative before performing other configurations that the ones described below.

The service.properties configuration file

Property files allow the Service Programs to be configured. These user editable properties are located in a special resource file named **service.properties** located in the Central Configuration database.

Important Entries within this file have precedence over entries having the same name in other resource files.

Every entry is preceded by a prefix in order to determine which program and which resource files an entry belongs to.

For example:

```
Getter.dataDir=..\data
Putter.dataDir=..\out
```

This entry defines the data directory for the Getter with

```
"..\data"
```

and for the Putter with

```
"..\out"
```

Keys that are end with (f) can be defined with the following formula:

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

Where:

key is from the hashtable,

delm is one of the "|", "?" or ":" characters and

function is one of the defined functions similar to the formulas in the properties files.

Example

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

The following table lists the possible prefixes used for the configuration of the Service Programs:

Prefix	Program
AL.	Account Loader
IL.	Image Loader
SRF.	Signature Reference Filter
XL.	XML Loader
Getter.	Getter
Putter.	Putter
DFP.	Day's Final Processing
F3.	F3-Loader
Crop.	Getter, SRF
DVA.	DataViever AccountLoader
DVIL.	DataViever ImageLoader
DVSRF.	DataViever SRF
DVG.	DataViever Getter

The following table lists the program specific configuration keys with their description:

Program	Key	Description
AccountLoader	AL.dataDir	Initial name of the data directory
AccountLoader	AL.logFile	Initial name of the log file
AccountLoader	AL.saveLog	Save the log messages into a log file?

Program	Key	Description
		1 - yes 0 - no
AccountLoader	AL.traceLevel	Trace level, refer to section Trace levels
AccountLoader	AL.dataSuffix	Suffix of the data files
AccountLoader	AL.rejectSuffix(f)	If rejectSuffix is not empty, then all records that could not be properly processed are written to a reject file. This file has the name of the data file and the extension ".rejectSuffix". Default: <empty>
AccountLoader	AL.renameSuffix(f)	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. No default.
AccountLoader	AL.renamePath	Specifies the directory of the data file to be moved after processing
AccountLoader	AL.report	Should a report be written? Possible values: yes, no Default: no
AccountLoader	AL.reportOnlyDB	Should only those records be reported that caused database changes? Possible values: yes, no Default: yes
AccountLoader	AL.ReportPath	Specifies the directory where report files are written Default: the current directory
AccountLoader	AL.workSuffix	Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same file name, but with workSuffix extension. This allows more than one program to run on the same data directory without processing the same file twice. Default: .wrk
AccountLoader	AL.errorSuffix(f)	Suffix for the error file. When the processing of the data file fails, the log is written to the file with the name of the data file with this extension. Default: .err

Program	Key	Description
AccountLoader	AL.deleteDataFile(f)	if set to true, the successfully processed data file is not renamed, but deleted. If an activation file exists, it is also deleted. Default: false
AccountLoader	AL.startNow	Should the start button be pressed automatically after the program is invoked? Possible values: yes, no
AccountLoader	AL.wait	Time (in seconds) to wait after processing all data files before searching for new data files. 0 - means no wait, but STOP after processing all data files Default: 10
AccountLoader	AL.ASV	Signatory enabled for ASV? 1 - yes 0 - no
AccountLoader	AL.QUALITY	Quality of this signature good enough for ASV? 1 - yes 0 - no
AccountLoader	AL.MAX_SIZE_SIC_M	Defines the maximum size of a mono signature. Bigger signatures are not processed. Default is the maximum size of the database column.
AccountLoader	AL.ruleCreate	Should a RULE record be created when creating a SIGNATORY record? Possible values: yes, no Default: no
AccountLoader	AL.rulePower	The POWER of a RULE record to be created when creating a SIGNATORY record. Possible values: S, C, N, T S - single right C - collective right (ANY two) N - no right T - all together right Default: S

Program	Key	Description
AccountLoader	AL.uniquePID	A check is taking place during creation of a signatory whether the given PersonalID already exists. The signatory is only created if the given PersonalID does not exist yet. 1 - yes 0 - no
ImageLoader	IL.dataDir	Initial name of the data directory
ImageLoader	IL.logFile	Initial name of the log file
ImageLoader	IL.saveLog	Save the log messages into a log file? 1 - yes 0 - no
ImageLoader	IL.traceLevel	Trace level, refer to section Trace levels
ImageLoader	IL.dataSuffix	Suffix of the data files
ImageLoader	IL.rejectSuffix(f)	If rejectSuffix is not empty, then all records that could not be properly processed are written to a reject file. This file has the name of the data file and the extension ".rejectSuffix". Default: .ilrej
ImageLoader	IL.renameSuffix(f)	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. Default: .ildone
ImageLoader	IL.renamePath	Specifies the directory of the Data file to be moved after processing
ImageLoader	IL.report	Should a report be written? Possible values: yes, no Default: yes
ImageLoader	IL.reportOnlyDB	Should only those records be reported that caused database changes? Possible values: yes, no Default: no
ImageLoader	IL.ReportPath	Specifies the directory where report files are written Default: the current directory

Program	Key	Description
ImageLoader	<code>IL.workSuffix</code>	<p>Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same filename, but with <code>workSuffix</code> extension. This allows more than one program to run on the same data directory without processing the same file twice.</p> <p>Default: <code>.wrk</code></p>
ImageLoader	<code>IL.errorSuffix(f)</code>	<p>Suffix for the error file. When the processing of the data file fails, the log is written to the file with the name of the data file with this extension.</p> <p>Default: <code>.err</code></p>
ImageLoader	<code>IL.deleteDataFile(f)</code>	<p>If set to true, the successfully processed data file is not renamed, but deleted.</p> <p>Default: <code>false</code></p>
ImageLoader	<code>IL.startNow</code>	<p>Should the start button be pressed automatically after the program is invoked?</p> <p>Possible values: yes, no</p>
ImageLoader	<code>IL.wait</code>	<p>Time (in seconds) to wait after processing all data files before searching for new data files.</p> <p>0 - means no wait, but STOP after processing all data files</p> <p>Default: 10</p>
ImageLoader	<code>IL.check.prePADCheck</code>	<p>Perform a pre-PAD check.</p> <p>1 - yes 0 - no</p> <p>For performance reasons it is possible to add a pre-PAD-check. PADs are recorded in <code>SC_RESULT</code>.</p> <p>Additional keys</p> <p>Minimum size for a PAD <code>IL.check.PADCleanedSizeMin</code></p> <p>Maximum size for a PAD <code>IL.check.PADCleanedSizeMax</code></p>
ImageLoader	<code>IL.check.PADCheck</code>	<p>Perform a PAD check.</p> <p>1 - yes 0 - no</p>

Program	Key	Description
		<p>PADs are recorded in SC_RESULT.</p> <p>Additional keys</p> <p>IL.PADLevel (match rate of PAD, 0-100)</p> <p>Defined text search for engines:</p> <p>IL.PADkey.0 = total number of PAD keystings</p> <p>IL.PADkey.1 = Signatory for</p> <p>IL.PADkey.2 = Signature on file</p> <p>IL.PADkey.3 = No Signature required</p> <p>IL.PADkey.4 = Verbally Authorized</p> <p>IL.PADkey.5 = Your depositor has authorized</p> <p>IL.PADkey.6 = Authorized signatory for</p> <p>IL.PADkey.7 = Payment has been authorized</p> <p>IL.PADkey.8 = Debit has been authorized</p> <p>IL.PADkey.9 = Check has been authorized</p> <p>IL.PADkey.10 = Has been pre-authorized</p> <p>(... more keys can be defined)</p>
ImageLoader	IL.PADEngine	<p>Name of the engine that performs the PAD-check.</p> <p>Possible values:</p> <ul style="list-style-type: none"> - GIA - none (no PAD engine available) <p>Default: GIA</p>
ImageLoader	IL.check.AmountRange	<p>Checks with an amount inside at least one of the specified amount ranges are excluded from processing.</p> <p>1 - yes</p> <p>0 - no</p> <p>Default: 1</p> <p>Additional keys</p> <p>For private accounts:</p> <p>IL.ignoreRangePrivate.0 = <number of ranges></p>

Program	Key	Description
		<p>IL.ignoreRangePrivate.1 = <range1> IL.ignoreRangePrivate.2 = <range2> ... (...more keys can be defined)</p> <p>For corporate accounts: IL.ignoreRangeCorporate.0 = <number of ranges> IL.ignoreRangeCorporate.1 = <range1> IL.ignoreRangeCorporate.2 = <range2> ... (...more keys can be defined)</p> <p>For other accounts: IL.ignoreRangeOther.0 = <number of ranges> IL.ignoreRangeOther.1 = <range1> IL.ignoreRangeOther.2 = <range2> ... (...more keys can be defined)</p> <p>Default: IL.ignoreRange.0 = <number of ranges> IL.ignoreRange.1 = <range1> IL.ignoreRange.2 = <range2> ... (...more keys can be defined)</p> <p>A range consists of 2 numbers, minimum and maximum (in cent), separated by comma.</p>
ImageLoader	IL.check.IRDCheck	<p>Checks for IRD. IRDs are excluded from processing.</p> <p>1 - yes 0 - no Default: 1</p> <p>Additional keys IL.SRFstoreIRD = 0 Possible values: 0, 1 1 - IRDs are used when creating input data for SRF</p>

Program	Key	Description
		0 - they will be omitted Default: 0
ImageLoader	IL.check.CorrectionItemCheck	Checks for Correction Item. 1 - yes 0 - no Default: 1 Correction Items are excluded from processing.
ImageLoader	IL.check.UnusualSizeCheck	Checks for unusual Size of the check. 1 - yes 0 - no Default: 1 Checks with unusual Size are excluded from processing.
ImageLoader	IL.check.ASVCheck	Check the ASV flag of the ACCOUNT. 1 - yes 0 - no Default: 1 In case of IL.check.ASVCheck=0 it is assumed that ACCOUNT.ASV=1. ACCOUNT with ASV=0 are considered as "blocked", Images for this account are excluded from processing.
ImageLoader	IL.check.SerialNoCheck	Checks the serial number of the check. If the serial number is less than a given minimum, the check is excluded from processing 1 - yes 0 - no Default: 1 Additional keys IL. check.minSerialNo=<minimum number> Default: 101
ImageLoader	IL.check.EngineCheck	Should new Images be compared with the existing images of the customer, using an engine? 0 - no (all images are processed)

Program	Key	Description
		1 - yes (only images not similar to existing images are processed). Default: 1
ImageLoader	IL.check.CreateSRFData	Should the ImageLoader create data for the SRF? 1 - yes 0 - no Default: 1 Additional keys IL.SRFTempSuffix - temporary file extension for SRF data. IL.SRFDataSuffix - final file extension for SRF data. The file name remains unchanged. IL.check.VariantsCheck - Should the ImageLoader check the count of existing variants to produce only SRF data when the maximum variant count is not yet reached? 1 - yes 0 - no Default: 1
ImageLoader	IL.maxImagesPrivate	Number of Images per signatory for private accounts
ImageLoader	IL.maxImagesCorporate	Number of Images per signatory for corporate accounts
ImageLoader	IL.maxImagesOther	Number of Images per signatory for other type of accounts
ImageLoader	IL.maxImages	Default Number of Images per signatory
ImageLoader	IL.minAgeOldImages	Minimum age of old checkstock images
ImageLoader	IL.deleteOldImages	Delete old chkeckstock images. Default: 0
SRF	SRF.dataDir	Initial name of the data directory
SRF	SRF.logFile	Initial name of the log file
SRF	SRF.saveLog	Save the log messages into a log file? 1 - yes 0 - no

Program	Key	Description
SRF	SRF.traceLevel	Trace level, refer to section Trace levels
SRF	SRF.user	The FraudOne user name to log on to the SignBase Server
SRF	SRF.password	The password of the FraudOne user to log on to the SignBase Server
SRF	SRF.dataSuffix	Suffix of the data files
SRF	SRF.rejectSuffix(f)	If rejectSuffix is not empty, then all records that could not be properly processed are written to a reject file. This file has the name of the data file and the extension ".rejectSuffix". Default: .srfrej
SRF	SRF.renameSuffix(f)	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. Default: .da_
SRF	SRF.renamePath	Specifies the directory of the data file to be moved after processing
SRF	SRF.report	Should a report be written? Possible values: yes, no Default: yes
SRF	SRF.reportOnlyDB	Should only those records be reported that caused database changes? Possible values: yes, no Default: no
SRF	SRF.ReportPath	Specifies the directory where report files are written. Default: the current directory
SRF	SRF.workSuffix	Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same filename, but with workSuffix extension. This allows more than one program to run on the same data directory without processing the same file twice. Default: .srfwrk
SRF	SRF.errorSuffix(f)	Suffix for the error file. When the processing of the data file fails, the log is

Program	Key	Description
		written to the file with the name of the data file with this extension. Default: .err
SRF	SRF.startNow	Should the start button be pressed automatically after the program is invoked? Possible values: yes, no
SRF	SRF.wait	Time (in seconds) to wait after processing all data files before searching for new data files. 0 - means no wait, but STOP after processing all data files Default: 10
SRF	SRF.createCustomer	Should a customer be created for the current variant, if this customer does not yet exist? Possible values: yes, no Default: no
SRF	SRF.createAccount	Should an account be created for the current variant, if this account does not yet exist? Possible values: yes, no Default: no
SRF	SRF.createDummySignatory	Should a so-called dummy signatory be created for the current variant, so far the customer has no dummy signatory and the new variant assigned to it? Possible values: yes, no Default: no
SRF	SRF.assignToDummySignatory	Should a so-called dummy signatory be created for the current variant, so far the customer has no dummy signatory? Possible values: yes, no Default: no
SRF	SRF.sivalMatchRate	Maximum match rate for new variants in comparison to all existing signatures of the customer. A new variant with a higher match rate will not be added. Match rates go from F5 (lowest) to AA (highest). Default: B5

Program	Key	Description
SRF	SRF.cleanLevel	Cleaning Level for the Sival Cleaning of a new signature. Values go from 0 to 1000. The higher the level the more black pixels are removed. A value of -1 disables the whole cleaning process. Default: 980
SRF	SRF.cleanLines	Perform a removing of lines before cleaning? 1 - yes 0 - no Default: 1
SRF	SRF.Quality.SmallSnippet	The maximum width for "Small" signatures. Signatures with a bigger width are considered as "Large". Default: all signatures are "Large"
SRF	SRF.Quality.SmallBBB	The maximum percentage of black pixels for small signatures. If the signature area has more, no variant will be created. Default: 100
SRF	SRF.Quality.SmallSimplicity	The maximum (Sival)simplicity of a small signature. 0 - not simple 100 - most simple If the signature is too simple, no variant will be created. Default: 100
SRF	SRF.Quality.LargeBBB	The maximum percentage of black pixels for large signatures. If the signature area has more, no variant will be created. Default: 100
SRF	SRF.Quality.LargeSimplicity	The maximum (Sival)simplicity of a large signature. 0 - not simple 100 - most simple If the signature is too simple, no variant will be created. Default: 100
XML-Loader	XL.dataDir	Initial name of the data directory
XML-Loader	XL.logFile	Initial name of the log file

Program	Key	Description
XML-Loader	XL.saveLog	Save the log messages into a log file? 1 - yes 0 - no
XML-Loader	XL.traceLevel	Trace level, refer to section Trace levels
XML-Loader	XL.user	The FraudOne user name to log on to the SignBase Server
XML-Loader	XL.password	The password of the FraudOne user to log on to the Signbase Server
XML-Loader	XL.dataSuffix	Suffix of the data files
XML-Loader	XL.rejectSuffix(f)	If rejectSuffix is not empty, then all records that could not be properly processed are written to a reject file. This file has the name of the data file and the extension ".rejectSuffix". Default: <empty>
XML-Loader	XL.renameSuffix(f)	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. Default: .da_
XML-Loader	XL.renamePath	Specifies the directory of the data file to be moved after processing
XML-Loader	XL.ReportPath	Specifies the directory where report files are written Default: the current directory
XML-Loader	XL.workSuffix	Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same filename, but with workSuffix extension. This allows more than one program to run on the same data directory without processing the same file twice. Default: .wrk
XML-Loader	XL.errorSuffix(f)	Suffix for the error file. When the processing of the data file fails, the log is written to the file with the name of the data file with this extension. Default: .err
XML-Loader	XL.startNow	Should the start button be pressed automatically after the program is invoked?

Program	Key	Description
		Possible values: yes, no
XML-Loader	XL.wait	Time (in seconds) to wait after processing all data files before searching for new data files. 0 means no wait, but STOP after processing all data files. Default: 10
XML-Loader	XL.minFileAge	minimum age of a file to be processed in the form <number><unit> where unit is one of: - s (second) - m (minute) - h (hour) - d (day) - w (week) Default: 0s (age is not relevant)
XML-Loader	XL.maxFileAge	maximum age of a file to be processed in the form <number><unit> where unit is one of: - s (second) - m (minute) - h (hour) - d (day) - w (week) Default: 0s (age is not relevant)
XML-Loader	XL.retrySuffix	If not empty then the file extension of files whose processing failed and it is worth to retry processing. Default: empty
XML-Loader	XL.retryInterval	The time span between the unsuccessful processing and the retry. Default: 1h
XML-Loader	XL.retryTimeout	The time span after the first unsuccessful processing where a retry should happen. After this time span the file will be finally deleted Default: 1w
XML-Loader	XL.fileSort	Sort criterion for processing data files: 0 - no sort

Program	Key	Description
		1 - alphanumeric 2 - date 3 - random classname – a project-specific class does the sort Default: 0
Getter	<code>Getter.dataDir</code>	Initial name of the data directory
Getter	<code>Getter.logFile</code>	Initial name of the log file
Getter	<code>Getter.saveLog</code>	Save the log messages into a log file? 1 - yes 0 - no
Getter	<code>Getter.traceLevel</code>	Trace level, refer to section Trace levels
Getter	<code>Getter.dataSuffix</code>	Suffix of the data files
Getter	<code>Getter.renameSuffix(f)</code>	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. Default: .done
Getter	<code>Getter.workSuffix</code>	Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same filename, but with workSuffix extension. This allows more than one program to run on the same data directory without processing the same file twice. Default: .wrk
Getter	<code>Getter.errorSuffix(f)</code>	Suffix for the error file. When the processing of the data file fails, the log is written to the file with the name of the data file with this extension. Default: .err
Getter	<code>Getter.startNow</code>	Should the start button be pressed automatically after the program is invoked? Possible values: yes, no
Getter	<code>Getter.wait</code>	Time (in seconds) to wait after processing all data files before searching for new data files. 0 - means no wait, but STOP after processing all data files.

Program	Key	Description
		Default: 10
Getter	Getter.PADEngine	<p>Name of the engine that performs the PAD-check. Possible values are:</p> <ul style="list-style-type: none"> - GIA - none (no PAD engine available) <p>Default: GIA</p>
Getter	Getter.check.PADCheck	<p>Perform a PAD check.</p> <p>1 - yes</p> <p>0 - no</p> <p>PADs are recorded in SC_RESULT.</p> <p>Additional keys</p> <p>Getter.PADLevel (minimum match rate of PAD, 0-100)</p> <p>Getter.AccountHolderLevel (match rate of Payor, 0-100)</p> <p>Getter.BlacklistLevel (match rate of Payee, 0-100)</p> <p>Defined text search for engines:</p> <p>Getter.PADkey.0=total number of PAD keystings</p> <p>Getter.PADkey.1=Signatory for</p> <p>Getter.PADkey.2=Signature on file</p> <p>Getter.PADkey.3=No Signature required</p> <p>Getter.PADkey.4=Verbally Authorized</p> <p>Getter.PADkey.5=Your depositor has authorized</p> <p>Getter.PADkey.6= Authorized signatory for ...</p> <p>Getter.PADkey.7=Payment has been authorized</p> <p>Getter.PADkey.8=Debit has been authorized</p> <p>Getter.PADkey.9=Check has been authorized</p> <p>Getter.PADkey.10= Has been pre-authorized</p> <p>(...more keys can be defined)</p>

Program	Key	Description
Getter	Getter.check.AmountRange	<p>Checks with an amount inside at least one of the specified number of amount ranges are recorded in SC_RESULT.</p> <p>1 - yes 0 - no Default: 1</p> <p>Additional keys Getter.ignoreRange.0=<number of ranges> Getter.ignoreRange.1=<range1> Getter.ignoreRange.2=<range2> (...more keys can be defined) A range consists of 2 numbers, minimum and maximum (in cent), separated by comma.</p>
Getter	Getter.check.IRDCheck	<p>Checks for IRD.</p> <p>1 - yes 0 - no Default: 1</p> <p>In case of IRD an additional SC_RESULT record will be created.</p>
Getter	Getter.check.CorrectionItemCheck	<p>Checks for Correction Item.</p> <p>1 - yes 0 - no Default: 1</p> <p>In case of a Correction Item an additional SC_RESULT record will be created.</p>
Getter	Getter.check.UnusualSizeCheck	<p>Checks for unusual Size of the check.</p> <p>1 - yes 0 - no Default: 1</p> <p>In case of an unusual Size an additional SC_RESULT record will be created.</p>
Getter	Getter.check.AccountCheck	<p>Checks for existence of the account.</p> <p>1 - yes 0 - no</p> <p>If the account does not exist an additional SC_RESULT record will be created.</p>

Program	Key	Description
Getter	<code>Getter.check.SignatureSizeCheck</code>	Checks the size of the signature. 1 - yes 0 - no Default: 1 If the size of the signature exceeds the maximum size of the appropriate database column an additional SC_RESULT record will be created.
Getter	<code>Getter.WFClient</code>	The Getter program accesses the Workflow to insert new items into the database. This property has to be set to true starting with Release 3.9. Otherwise items inserted by the Getter will not be processed.
Getter	<code>Getter.Host</code>	The name of the workstation hosting the workflow process. The Getter tries to connect to this host to get a connection with the Workflow process in order to insert items into the database. Since Release 3.9 this is a mandatory property. Default: localhost.
Getter	<code>Getter.Port</code>	The port number used for communication between the Getter and the Workflow process, e.g. <code>Getter.Port=2018</code> . This must be the same number as configured for the Workflow in SrvMngr4.ini (see WFPort).
Getter	<code>Getter.Timeout</code>	The timeout till the Getter waits for a response from the Workflow process. If the Workflow does not respond within this timeframe an error is reported in the log file.
Getter	<code>Getter.maxRetries</code>	The number of retries the Getter performs in case the Workflow does not respond. If the number of retries is reached the Getter stops processing.
Putter	<code>Putter.dataDir</code>	Initial name of the data directory
Putter	<code>Putter.logFile</code>	Initial name of the log file
Putter	<code>Putter.saveLog</code>	Save the log messages into a log file? 1 - yes 0 - no
Putter	<code>Putter.traceLevel</code>	Trace level, refer to section Trace levels

Program	Key	Description
Putter	<code>Putter.dataSuffix</code>	Suffix of the data files
Putter	<code>Putter.startNow</code>	Should the start button be pressed automatically after the program is invoked? Possible values: yes, no
Putter	<code>Putter.wait</code>	Time (in seconds) to wait after processing all unprocessed records in the database before searching for new unprocessed data. 0 - means no wait, but STOP after processing all data Default: 10
Putter	<code>Putter.writeCustomerType</code>	Has the CustomerType of an check to be written in the result file? (This requires an additional SELECT-statement.) 1 - yes 0 - no
DFP	<code>DFP.dataDir</code>	Initial name of the data directory
DFP	<code>DFP.logFile</code>	Initial name of the log file
DFP	<code>DFP.saveLog</code>	Save the log messages into a log file? 1 - yes 0 - no
DFP	<code>DFP.traceLevel</code>	Trace level, refer to section Trace levels
DFP	<code>DFP.startNow</code>	Should the start button be pressed automatically after the program is invoked? Possible value: yes, no
DFP	<code>DFP.Days</code>	Possible values: -1, 0, >0 -1 - all entries are deleted >0 - all processed entries older than "Days" are deleted =0 - all processed entries are deleted, however, dependent on WaitForFR2 and DelProcDate
DFP	<code>DFP.WaitForFR2</code>	Possible values: yes, no yes - Records with FINAL_RESULT=2 are never considered to be processed. These records can only be deleted with Days!=0 no - FINAL_RESULT is ignored

Program	Key	Description
DFP	DFP.DelProcDate	<p>Possible values: -1, >=0</p> <p>>=0 - Means that records with PROC_DATE defined are considered to be processed only if <PROC_DATE + 'DelProcDate' days> is not in the future</p> <p>-1 - Means that PROC_DATE is ignored</p>
DFP	DFP.deleteFiles	<p>If true, additional to DFP's actual work, files are deleted.</p> <p>Default: false</p>
DFP	DFP.deleteFilesFirst	<p>If true, files are deleted before deleting any database tables, otherwise files are deleted after the database work.</p> <p>Default: false</p>
DFP	DFP.delPath.0	<p>The number of directories where files are to be deleted. The directories are named in the keys delPath.1, delPath.2 etc., analog the regular expressions and ages of the files.</p> <p>Default: 0</p>
DFP	DFP.delPath.1	<p>The first directory where files are to be deleted. If delPath.1 is empty, nothing will be deleted.</p> <p>Default: leer</p>
DFP	DFP.delPattern.1	<p>Regular expression to denote the files in directory delPath.1. If delPattern.1 leer, is empty, nothing will be deleted</p> <p>Example</p> <p>delPattern.1=\\.done\$</p> <p>removes all files in the directory ending with ".done".</p> <p>Default: leer</p>
DFP	DFP.delDays.1	<p>only those files that are at least "delDays.1" days old are deleted.</p> <p>Default: 0</p>
F3-Loader	F3.dataDir	Initial name of the data directory
F3-Loader	F3.logFile	Initial name of the log file
F3-Loader	F3.saveLog	<p>Save the log messages into a log file?</p> <p>1 - yes</p> <p>0 - no</p>

Program	Key	Description
F3-Loader	F3.traceLevel	Trace level, refer to section Trace levels
F3-Loader	F3.dataSuffix	Suffix of the data files
F3-Loader	F3.rejectSuffix(f)	If rejectSuffix is not empty, then all records that could not be properly processed are written to a reject file. This file has the name of the data file and the extension ".rejectSuffix". Default: <empty>
F3-Loader	F3.renameSuffix(f)	Suffix for the renaming of the data file. When specified, the data file's dataSuffix is replaced by renameSuffix after processing. No default.
F3-Loader	F3.renamePath	Specifies the directory of the data file to be moved after processing
F3-Loader	F3.report	Should a report be written? Possible values: yes, no Default: yes
F3-Loader	F3.reportOnlyDB	Should only those records be reported that caused database changes? Possible values: yes, no Default: yes
F3-Loader	F3.ReportPath	Specifies the directory where report files are written Default: the current directory
F3-Loader	F3.workSuffix	Suffix for working files. The data file is locked by creating an empty working file in the same directory, with the same filename, but with workSuffix extension. This allows more than one program to run on the same data directory without processing the same file twice. Default: No working file
F3-Loader	F3.errorSuffix(f)	Suffix for the error file. When the processing of the data file fails, the log is written to the file with the name of the data file with this extension. Default: .err
F3-Loader	F3.deleteDataFile(f)	If set to true, the successfully processed data file is not renamed, but deleted.

Program	Key	Description
		Default: false
F3-Loader	F3.startNow	Should the start button be pressed automatically after the program is invoked? Possible values: yes, no
F3-Loader	F3.wait	Time (in seconds) to wait after processing all data files before searching for new data files. 0 - means no wait, but STOP after processing all data files. Default: 10
F3-Loader	F3.CountryId	Country Id Default: 840
F3-Loader	F3.BankCode	Bank Code Default:0
F3-Loader	F3.blockVarValidFrom	Count of days beginning with the processing day when the blocked Variant will be valid Default: 0 days (immediately)
F3-Loader	F3.unblockVarValidFrom	Count of days beginning with the processing day when the unblocked Variant will be valid Default: 0 days (immediately)
F3-Loader	F3.blockImageValidFrom	Count of days beginning with the processing day when the blocked Image will be valid Default: 0 days (immediately)
F3-Loader	F3.unblockImageValidFrom	Count of days beginning with the processing day when the unblocked Image will be valid Default: 0 days (immediately)
F3-Loader	F3.blockingTime	Count of days beginning with the processing day when the blocked Image will be unblocked Default: 0 days (immediately)
All JDBC Programs	Database.URL	Database location. Syntax for Oracle database: jdbc:oracle:thin:pc_name:port:database_name Syntax for SQL Server database: jdbc:microsoft:sqlserver://pc_name:port Syntax for DB2 database: jdbc:db2:database_name

Program	Key	Description
All JDBC Programs	Database.propValue1	User Id
All JDBC Programs	Database.propValue2	Password
All JDBC Programs	Database.catalog	Catalog name
Getter, Putter, DFP	DatabaseSC.URL	SignCheck database location
Getter, Putter, DFP	DatabaseSC.propValue1	User Id Default: Database User Id
Getter, Putter, DFP	DatabaseSC.propValue2	Password Default: Database Password
Getter, Putter, DFP	DatabaseSC.catalog	Catalog name Default: Database Catalog name
DFP	DatabaseDWH.URL	SignCheck database location
DFP	DatabaseDWH.propValue1	User Id Default: Database User Id
DFP	DatabaseDWH.propValue2	Password Default: Database Password
DFP	DatabaseDWH.catalog	Catalog name Default: Database Catalog name
Getter, SRF	Crop.CROP0Default	Default rectangle for the search area where the signature 1 on the front side will be located. Default: Crop.CROP0Default=1,1,-1,-1
Getter, SRF	Crop.CROP1Default	Default rectangle for signature 1 on the front side. Default: Crop.CROP1Default=1,1,-1,-1
Getter, SRF	Crop.CROP2Default	Default rectangle for signature 2 on the front side. Default: Crop.CROP2Default=1,1,-1,-1
Getter, SRF	Crop.CROP3Default	Default rectangle for the signature on the backside. Default: Crop.CROP3Default=1,1,-1,-1

Program	Key	Description
Getter, SRF	Crop.CROP1-Itemtype	<p>Rectangle for signature 1 on the front side for Itemtype Itemtype.</p> <p>The Itemtype results from the values of C_TXN, FORM_TYPE and FORM_TEXT_CODE.</p> <p>This entry should be repeated for every Itemtype whose rectangle differs from the default rectangle.</p> <p>Default: Crop.CROP1Itemtype=Crop.CROP1Default</p>
Getter, SRF	Crop.CROP2-Itemtype	<p>Rectangle for signature 2 on the front side for Itemtype Itemtype.</p> <p>The Itemtype results from the values of C_TXN, FORM_TYPE and FORM_TEXT_CODE.</p> <p>This entry should be repeated for every Itemtype whose rectangle differs from the default rectangle.</p> <p>Default: Crop.CROP2Itemtype=Crop.CROP1Default</p>
Getter, SRF	Crop.CROP3-Itemtype	<p>Rectangle for the signature on the backside for Itemtype Itemtype.</p> <p>The Itemtype results from the values of C_TXN, FORM_TYPE and FORM_TEXT_CODE.</p> <p>This entry should be repeated for every Itemtype whose rectangle differs from the default rectangle.</p> <p>Default: Crop.CROP3Itemtype=Crop.CROP3Default</p>
Getter, SRF	Crop.CROP0Checktype	<p>Only for US banks</p> <p>A Rectangle defining the search area where the signature 1 on the front side will be located. The Checktype results from the size of the check and of FORM_TYPE.</p> <p>Possible Checktypes are:</p> <p>Retail – a check with $1360 \leq \text{width} < 1480$ and $570 \leq \text{height} \leq 680$</p> <p>CorporateHighLow– a check with $1950 < \text{width} \leq 2100$ and $650 \leq \text{height} < 750$</p> <p>CorporateHighHigh– a check with $1950 < \text{width} \leq 2100$ and $750 \leq \text{height} \leq 960$</p> <p>CorporateLowLow– a check with $1720 \leq \text{width} \leq 1950$ and $650 \leq \text{height} < 750$</p>

Program	Key	Description
		<p>CorporateLowHigh– a check with 1720<=width<=1950 and 750<=height<=870</p> <p>CIRetail CICorporateHighLow CICorporateHighHigh CICorporateLowLow CICorporateLowHigh IRD</p> <p>- The prefix CI indicates that the check is a "Correction Item"</p> <p>- IRD stands for "Image Replacement Document"</p> <p>Default: the value of Crop.CROP0Default</p>
Getter	Crop.CROPChecktype	<p>Only for US banks</p> <p>A Rectangle for the signature 1 on the front side for the case that the search for this signature failed.</p> <p>The Checktype results from the size of the check and of FORM_TYPE. (see above)</p> <p>Default: the value of Crop.CROP1Default</p>
Getter, ImageLoader, SRF	Getter.retailMinX and IL.retailMinX and SRF.retailMinX resp. or Crop.retailMinX	<p>The minimum width for a Retail check</p> <p>Default: 1360</p>
Getter, ImageLoader, SRF	Getter.retailMaxX and IL.retailMaxX and SRF.retailMaxX resp. or Crop.retailMaxX	<p>The maximum width for a Retail check</p> <p>Default: 1480</p>
Getter, ImageLoader, SRF	Getter.retailMinY and	<p>The minimum height for a Retail check</p> <p>Default: 570</p>

Program	Key	Description
	IL.retailMinY and SRF.retailMinY resp. or Crop.retailMinY	
Getter, ImageLoader, SRF	Getter.retailMaxY and IL.retailMaxY and SRF.retailMaxY resp. or Crop.retailMaxY	The maximum height for a Retail check Default: 680
Getter, ImageLoader, SRF	Getter.corporateLowMinX and IL.corporateLowMinX and SRF.corporateLowMinX resp. or Crop.corporateLowMinX	The minimum width for a small Corporate check Default: 1720
Getter, ImageLoader, SRF	Getter.corporateLowMaxX and IL.corporateLowMaxX and SRF.corporateLowMaxX resp. or Crop.corporateLowMaxX	The maximum width for a small Corporate check Default: 1950
Getter, ImageLoader, SRF	Getter.corporateLowMinY and IL.corporateLowMinY and SRF.corporateLowMinY resp. or Crop.corporateLowMinY	The minimum height for a small Corporate check Default: 650
Getter, ImageLoader, SRF	Getter.corporateLowMaxY and IL.corporateLowMaxY and	The maximum height for a small Corporate check Default: 750

Program	Key	Description
	SRF.corporateLowMaxY resp. or Crop.corporateLowMaxY	
Getter, ImageLoader, SRF	Getter.corporateLowHighMaxY and IL.corporateLowHighMaxY and SRF.corporateLowHighMaxY resp. or Crop.corporateLowHighMaxY	The maximum height for a small Corporate check with big height Default: 870
Getter, ImageLoader, SRF	Getter.corporateHighMinX and IL.corporateHighMinX and SRF.corporateHighMinX resp. or Crop.corporateHighMinX	The minimum width for a big Corporate check Default: 1950
Getter, ImageLoader, SRF	Getter.corporateHighMaxX and IL.corporateHighMaxX and SRF.corporateHighMaxX resp. or Crop.corporateHighMaxX	The maximum width for a big Corporate check Default: 2100
Getter, ImageLoader, SRF	Getter.corporateHighMinY and IL.corporateHighMinY and SRF.corporateHighMinY resp. or Crop.corporateHighMinY	The minimum height for a big Corporate check Default: 750
Getter, ImageLoader, SRF	Getter.corporateHighMaxY and IL.corporateHighMaxY and SRF.corporateHighMaxY	The maximum height for a big Corporate check Default: 960

Program	Key	Description
	resp. or <code>Crop.corporateHighMaxY</code>	
Getter, SRF	<code>Crop.clipResolution</code>	The assumed resolution for the cropping values above. If the resolution of the check differs, the cropping values are recalculated. Default: 240
SRF	<code>Variants.maxVariants</code>	Number of Variants per signatory
SRF	<code>Variants.maxVariantsPrivate</code>	Number of Variants per signatory for private accounts
SRF	<code>Variants.maxVariantsCorporate</code>	Number of Variants per signatory for corporate accounts
SRF	<code>Variants.maxVariantsOther</code>	Number of Variants per signatory for other type of accounts
SRF	<code>Variants.validFrom</code>	Number of days until variant becomes valid
SRF	<code>Variants.minAgeOldVariants</code>	Minimum age of old variants
SRF	<code>Variants.deleteOldVariants</code>	Delete all old variants. 1 - yes 0 - no Default: 1
Data Viewer	<code>DVxxx.dataDir</code> (xxx stands for A, IL, G, SRF)	Data directory name
Data Viewer	<code>DVxxx.logFile</code>	Log file name
Data Viewer	<code>DVxxx.saveLog</code>	Save the log messages into a log file? 1 - yes 0 - no
Data Viewer	<code>DVxxx.traceLevel</code>	Trace level, refer to section Trace levels

Important The **service.zip** file, located in the class path includes initial environment installation configuration. Consult your Kofax support representative before making any changes to this file.

Report Files

The FraudOne core functionality allows for the creation of report files for the Account Loader, Image Loader, Signature Reference Filter, XML Loader and Fraud Feedback File Loader. Report files record the actions and database changes made by the Service Programs. The reports can consist of one or more report data lines sorted in alphabetical order. The format and description of the data entries are defined in one of the properties files contained in **service.zip**.

The file name and extension are configurable. The fields are separated by delimiters, which enable a simple import into reporting tools. Each service program has a different report file format but with the similar **‘.rep’** extension.

Controlled by a property flag, the reporting interface is configured by default to only report activities resulting in reference database changes. As a result the reason codes for rejected items are not included.

The following are the predefined keys of the Service Programs:

Key	Default	Description
REPORT	0	1 - if a report is to be created, otherwise 0
REPORT.DB	1	1 - if only database changes are to be reported 0 - all activities are reported
REPORT.DELIMITER	","	Comma - Delimiting character for the fields of one report line

Account Loader Report File

For every line resulting in a database transaction (update/delete/insert) the Account Loader writes a report line.

The report file name is based on Account Loader process identifier number (if available) and the processing date. **"ALnn_yyyymmdd.rep"**.

The reporting output follows the following format, all fields are delimited by ";".

Field	Length (Char)	Description
BNO	3	Bank number
Customer	1-20	Customer number
Action	1	I - Inserted U - Updated P - Purged

Image Loader Report File

The Image Loader logs all transactions to enable database progression analysis based on the contents of the report files.

The report file name is based on Image Loader process identifier number (if available) and the processing date. "IRLnn_yyyymmdd.rep".

The reporting output follows the following format, all fields are delimited by ";".

Field	Length (Char)	Description
BNO	3	Bank number
Customer	14	Customer number
Type	1	Customer Type
DocID	20	Unique document ID
NumImg	2	Number of incumbent reference check images
Action	1	I - new reference inserted C - no reference inserted - account complete Q - no reference inserted - bad quality R - no reference inserted - item image considered risky P - reference parameter inserted (check image only) N - not inserted due to other reasons S - not selected for insert J - rejected A - amount too little T - image too similar to an existing one B - missing SignBase data V - Account blocked
ImgNo	1	Reference image number of new inserted image. (not added => " ")
Match	4	Best match rate against incumbent reference " " = no incumbent reference or no validation performed
MatchImg	1	Reference number for above match rate

The following fields are written only if 'full report' is enabled (reportOnlyDB=false):

Field	Length (Char)	Description
Quality	2	00 - OK 01 - Unusual size 02 - PAD detected 03 - IRD

Field	Length (Char)	Description
		04 - serial number too small or account complete 05 - no Image 06 - correction Item 07 - bad resolution
Test	4	Confidence level from PAD-test for check images. " " = no test

Signature Reference Filter Report File format

The Signature Reference Filter logs all transactions enabling database progression analysis based on the contents of the report files.

The report file name is based on Image Loader process identifier number (if available) and the processing date **"SRFnn_yyyymmdd.rep"**.

The reporting output follows the following format, all fields are delimited by ";".

Field	Length (Char)	Description
BNO	3	Bank number
Customer	14	Customer number
Type	1	Customer Type
DocID	20	Unique document ID
NumSign	2	Number of incumbent reference signatures
Action	1	I - new reference inserted C - no reference inserted - account complete Q - no reference inserted - bad quality R - no reference inserted - item image considered risky N - not inserted due to other reasons S - not selected for insert J - rejected A - amount too little T - signature too similar to an existing one B - missing SignBase data D - item not inserted because it is an IRD

Field	Length (Char)	Description
SignNo	1	Reference signature number of new inserted signature (not added => " ")
Match	4	Best match rate against incumbent reference signature " " - no incumbent reference signature or no validation performed
MatchSign	1	Reference signature number for above match rate

The following fields are written only if full reporting is enabled (reportOnlyDB=false):

Field	Length (Char)	Description
Quality	2	00 - OK 01 - No image left after clipping white space 02 - Snippet height to low 03 - Snippet width to slim 04 - Too few pixels left in image 05 - Not enough parameters found in snippet 06 - Too much pixels left in image
Test	4	Snippet size from signature existence test for signatures. " " - no test

Fraud Feedback File Loader Report File format

For every line that resulted in a database transaction (update/delete/insert) the F3-Loader will write a report line with the following content:

The report file name is based on F3-Loader process identifier number (if available) and the processing date. "F3nn_yyyymmdd.rep".

The reporting output follows the following format, all fields are delimited by ";".

Field	Length (Char)	Description
BNO	3	Bank number
Customer	14	Customer number
DocID	20	Unique document ID
Reference-Type	1	S - Signature

Field	Length (Char)	Description
		I - Image A - Account
Action	1	D - Logically Deleted B - Blocked from usage

Trace levels

All Service Programs contain a log area where log information is written. Optionally, the log can be written to a file. The Trace Level defines the quantity of info lines desired in the log file. This is configured using the **service.properties** with the following syntax:

```
Serviceprogram.tracelevel value
```

Where the *value* is the sum of the Trace level desired.

Example

```
AL.tracelevel 11
```

In the example above, the values for INFO (8), WARNING (2), and ERROR (1) lines will be displayed.

The following table describes the Trace Level values.

Note Minimum TraceLevel = 11

Name	Value	AL/IL/F3	XML	SRF	G	P	DFP	Description
ERROR	1	x	x	x	x	x	x	Error messages
WARNING	2	x	x	x	x	x	x	Warnings
DEBUG	4	x	x	x	x	x	x	Information for tests
INFO	8	x	x	x	x	x	x	General information
RESOURCE	16	x	x	x	x	x	x	List contents of the property files
SQL	32	x	x		x	x	x	Database messages (SQL activities)
SUBSTITUTE	64	x	x	x	x	x	x	Step by step protocolling of substitution
SELECTION	128			x	x			Step by step protocolling of selection
PERFORMANCE	256	x	x	x	x	x	x	Performance information

Where:

AL = Account Loader

IL = Image Loader

F3 = Fraud Feedback File Loader

XML = XML Loader

SRF = Signature Reference Filter

G = Getter, **P** = Putter

DFP = Day's Final Processing

Note The minimum TraceLevel is 11 for all Service Programs.

This is the combination of the values 1 (ERROR), 2 (WARNING) and 8 (INFO), which is needed to assure that all important events are reported in the logfile and the graphical interface.

Work File processing

Working with Service Programs that process input files (Account Loader, Image Loader, F3-Loader, SRF, and Getter) makes it possible to define work files. For every Data file that is opened, a work file is created containing processing progress information.

The following advantages to work file processing are:

More than one service-program can work on the same data directory without processing a data file twice

If a service-program was stopped or interrupted, the data file processing can continue later at the last position of processing

Requirements for use of Work Files

In order to use the Work Files:

- The data directory must be writable. Otherwise, a writable work directory should be defined
- The processed data files must be renamed or deleted

After the complete processing of an input file, the associated work file is deleted. After a Service Program is interrupted the work file continues to provide information when it is activated later. For this reason, the work files should not be changed or deleted manually. The work file retains the same format and the same name as the data file, but an additional extension (*filename.dat* would get *filename.dat.wrk*).

Reject File processing

When using Account Loader, Image Loader, F3-Loader, and Signature Reference Filter, it is possible to write the unprocessed records to a reject file. The format of the records is not changed. Additionally, at least one comment line is written to this record providing the reason of the rejection. A separate reject file is created for every data file.

For example:

```
# line 3:
```

```
CA002840      0000005710004846      KELLNER CRAIG      0000005710004846
      USD      0
# COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0407N
      Assig...
```

Advantages of Reject Files

The following advantages of the Reject Files:

- Rejected records are better documented, providing a more effective investigation
- After the reason of the failure has been eliminated it possible to re-process these files by renaming the reject file.

Password encryption

Account Loader, Getter, Putter and the Day's Final Processing Service Programs require a valid UserID and password combination in order to access the SignBase / SignCheck database. The Signature Reference Filter also requires a valid UserID and password in order to access the SignBase server.

UserID and password are usually stored in the **SrvMngr.ini** or **service.properties** files. In order to limit access to non-authorized users, it is possible to encrypt the stored password. All Service Programs that require password authentication will first allow a login with the initial values. If access is denied, the Service Program assumes that the password is encrypted. It decrypts the password and attempts to log in again. If this fails, the user is prompted to enter a valid UserID and password. The encrypted password replaces the original password in the **SrvMngr.ini** or **service.properties** files.

Encrypting and decrypting requires the use of a FraudOne Licensing Key. Consult your Kofax business representative for further information regarding licensing.

Archive Interface Server

This chapter describes the FraudOne Archive Interface Server (AIS) in the standard functionality as well as the configurations that can be performed by an administrator. The availability of certain features in your installation may vary depending on the products you have purchased.

Overview Archive Interface Server

The FraudOne Archive Interface Server (AIS) is used for the indexing and transferring of images and XML data to the FraudOne system for the purpose of processing and visualization. The images may consist of account card images and other images of documents related to the account or the account holder (contractual documents, identity cards, or passport images). The AIS uses 'locators' to monitor local or external network drives and indexes the located data.

There are three available locators, the Image Locator, the XML Locator, and the Kleindienst XML Locator.

The Image Locator loads image files (i.e. TIFF, BMP) and can automatically assign images to accounts. For this purpose a naming convention

<AccountNo>.<UniquelImageID>.<ImageType>

has been defined.

The XML Locator loads well formed XML files based on the document type definition **signplus.dtd**. Should a new file be located which matches the search criteria for a 'locator' then the file will be registered for processing. Once registered for processing, the FraudOne client can load, process and store the located data in the FraudOne database. The **AIS** logs the information about the located, registered and processed data. The **AMS** (Archive Management Server) component of the AIS provides certain statistics about this process.

Archive Interface Server installation

The following components are required for the Archive Interface Server installation.

Database components

A JDBC driver installation is required for the database used. All CLASSPATH entries for the JDBC must be either entered in the user environment in the CLASSPATH or can be manually added to the CLASSPATH entry in the **ArchiveIFserver.cmd**.

Refer to the document *FraudOne Technical Specifications* for the required Versions for each database product.

DB2

Example for DB2:

```
CLASSPATH=.....C:\SQLLIB\java\db2java.zip; C:\SQLLIB\java\runtime.zip;
```

This entry must be modified to match the current DB2 installation.

When using DB2, the JDBC 2 driver must be activated before the initial use. This is done by starting a utility delivered by IBM in your current DB2 installation.

Example for DB2:

```
C:\SQLLIB\java12\usejdbc2.bat (for JDBC 2)
```

By starting this utility from IBM the runtime files will be set to use JDBC 2.0. You may also verify what is actually used by viewing the **'inuse'** (text file) which is also located in the same folder as the ***.bat** file. The **'inuse'** file will be created after setup for JDBC 2.x.

Note When installing IBM DB2 always install the full DB version. JDBC will not function correctly when only the client is installed. The database may be located externally, but the full software installation is required.

Oracle

Example for Oracle 10g:

```
CLASSPATH = ... ojdbc14.jar;
```


MS SQL Server

Example for SQL Server 2005:

```
CLASSPATH=... sqljdbc.jar;
```

Java Client components

All required components for the Java Client are also required for the Archive Interface Server. If the Java Client is installed on the same hardware platform then the CLASSPATH should be modified so that the zip's, jar's, etc. can be found.

Server components

The following server components are used by the Archive Interface Server and should be located in the same directory as the AIS:

- srvmngr4.exe
- srvmngr4.ini
- SrvmInitJava.exe

Starting the Archive Interface Server

The Archive Interface Server can be started either via the Server Manager or via command line or script.

Starting from the Server Manager

When the Archive Interface Server is started by the server manager the configuration is performed as prescribed in section [Archive Interface Server](#).

The instance name from the topography configuration will be used when loading properties.

The Archive Interface Server will also be visible in the Server Monitor.

Starting from the Command Line

It is also possible to start the Archive Server from a command file. See the sample file **ArchiveIFServer.cmd** delivered in your shipment.

Should the instance name for selecting the correct properties be required the following setting will be required. Since we have no configured instance name when starting from the command script as opposed to starting by the server manager the instance name can be passed in the command line as an option.

Note The classpath list for both input commands below have been shortened for visualization purposes and will not function. For a current sample for your installation see the **ArchiveIFServer.cmd** in your shipment.

Sample command line input with no optional instance name:

```
java -cp archive.jar;ams.jar; ... .;%CLASSPATH% -Xms32m -Xmx128m
Archive
```

Sample command line input with instance name:

```
java -cp archive.jar;ams.jar; ... .;%CLASSPATH% -Xms32m -Xmx128m
Archive AISName
```

The **AISName** in this sample case specifies the instance name used as with the section name when starting via the Server Manager.

Archive Interface Server configuration

The Archive Interface Server requires the use of various client components, server components, and property files.

All property files are maintained in the Central Configuration, files from the file system are no longer loaded. All required property settings must be performed by the Administration Client.

See section [Thin Client Property Files](#) for information regarding the property files used for the Archive Interface Server.

See section [Business Model Property Files](#) for information regarding the property files used for Business Model required by the Archive Interface Server.

All property files must be assigned to the 'InstanceType' - ArchiveServer.

custom.properties

The Archive Interface Server requires the identical information used for the Java Client configuration file **custom.properties** element located in the central configuration server.

The following table describes the key settings used in this property file:

```
BankNo =
CountryId = 049
BankCode =
```

Parameter	Client's Default Settings
BankNo =	For multiple-client capability: Bank number that will be entered as standard in search dialogs. BankNo 999 is not allowed!
CountryId = 049	Nationality key Default: 049
BankCode =	BankCode (such as routing code) that is entered as standard in search dialogs. Default: blank

sizes.properties

The **sizes.properties** configuration element is located in the private section of the configuration element which originates from the custom.zip file.

Note Consult your Kofax technical representative before making any changes to the contents of the **custom.zip** file.

The following parameters can be set in this property element:

```
Bankcode.size = 4
Bankcode.minsize = 3
Customerno.size = 5
Customerno.minsize = 5
Accountno.size = 5
Accountno.minsize = 5
Branchcode.size = 8
Branchcode.minsize = 0
```

archive.properties

The **archive.properties** configuration file contains items that are used by the Archive Interface Server and are not located in the Client property files. Entries allowing for multiple folders are entered as follows:

```
...=c:/tmp/images;X:/queue/images/new;
```

For entries which allow multiple file types they are entered as follows:

```
...=bmp;tif;
```

The following parameters can be set:

```
Server.ShowTrace =
Server.AllowDbManagement =
Server.AllowDbAdmin =
Server.UseHostDB =
Server.UseHostDB =
Server.TraceLevel = 11
Server.ConsoleEnabled =
Server.AcceptClientRequests =
ImageServer.MaxThreads =
UseProtocol =
AllowSbCreateCustomer =
MaxImagePackageSize =
Custom.AccountModel =
UseLockingForUser =
ElapsedLockedForUserTime =
UsePlaceOnHold =
ElapsedOnHoldTime =
SortGetNextBy =
SortGetNextOrder =
StatisticsInterval.FirstTime =
StatisticsInterval =
```

Image File Locator

```
ImageLocatorEnabled
ImageLocatorPath
ImageLocatorTypes
ImageLocatorPause
FormattedFileName
UseFileNameAsAccount
```

[GroupByFileName](#)
[ReplaceAllImagesOnUpdate](#)
[EmbeddedAccountFormat](#)
[EmbeddedAccountNoPlaceholder](#)
[AccountFormat](#)
[AccountNoDigitPlaceholder](#)
[WildcardPlaceholder](#)
[DeleteInvalidImageFiles](#)

SOFTPRO XML Locator

[XmlLocatorEnabled](#)
[XmlLocatorPath](#)
[XmlLocatorType](#)
[XmlLocatorPause](#)
[AppendImagesToXML](#)
[XMLComplimentsImages](#)

Locked Status

Parameter	Description
ElapsedStatusToReady =	This controls the monitoring of status entries in the status table. If enabled any locked images which were sent to a client but a confirmation was never received (before the specified timeout) the status will be sent to ready or done. 0 - Status will be reset to DONE 1 - Status will be reset to READY Default: 1
ElapsedStatusTime =	Time span (in minutes) in which a reply is expected from the requesting client regarding the status of a received image package. After this time has elapsed the locked images will be set either to READY or DONE as specified in 'ElapsedStatusToReady'. Default: 30

Completed Files

Parameter	Description
DeleteCompletedFiles =	Delete completed image files after confirmation received from client 0 - No deleting occurs. The removal of complete files must be performed externally. The status entry will also not be removed. 1 - The files will be deleted and the status entry removed. Default: 1
RunPerformanceTest =	Runs performance test. As soon as a file is processed the entry is removed from the status table but the file is not deleted.

Parameter	Description
	Default: OFF

Kleindienst XML Locator

Parameter	Description
KleindienstXmlLocatorEnabled =	<p>Activate the XML locator for Kleindienst XML files. If activated the specified folders will be monitored for new image file types as specified.</p> <p>0 - Off 1 - On Default: 1</p>
KleindienstLocatorPath =	<p>Specifies the folders to be monitored. All folder must be input fully qualified and end with ';'. If multiple folders are desired then the next folder begins after the delimiter (;).</p>
KleindienstLocatorPause =	<p>A pause time can be specified for the locator. After all XML files have been checked the locator will wait the specified time (in minutes) before restarting.</p> <p>Default: 10</p>
KleindienstUpdateDelay =	<p>After all files specified within the XML file have been registered the updater can be started after a specified delay in minutes. This will update the status of the registered images and cannot be turned off.</p> <p>This runs only one time. Any image status entry's not updated will then be rechecked when the 'ElapsedStatusTime' elapses.</p> <p>Default: 1 (minute)</p>
ValidateKleindienstXML =	<p>Specify whether the processed XML files from Kleindienst are to be validated or not. If so then the name of the DTD must be entered within the XML file and the correct DTD available.</p> <p>Default: 0</p>
DeleteKleindienstXML =	<p>Specify whether the processed XML from Kleindienst may be deleted or not. If not deleted then the file name will automatically be extended with „DONE.xxxx' (xxxx is the timestamp of completion).</p> <p>Default: 0</p>
UseKleindienstBlackFront =	<p>Up to 4 images can be specified within the Kleindienst XML file. If the black and white front image is desired (if available) set to '1', else to '0'.</p> <p>Default: 1</p>

Parameter	Description
UseKleindienstBlackRear =	Up to 4 images can be specified within the Kleindienst XML file. If the black and white rear image is desired (if available) set to '1', else to '0'. Default: 1
UseKleindienstGreyFront =	Up to 4 images can be specified within the Kleindienst XML file. If the grey front image is desired (if available) set to '1', else to '0'. Default: 1
UseKleindienstGreyRear =	Up to 4 images can be specified within the Kleindienst XML file. If the grey rear image is desired (if available) set to '1', else to '0'. Default: 1

ams.properties

The following table describes the key settings used in this property file:

```

Server.ShowTrace = 0
Server.TraceLevel = 11
Server.AcceptClientRequests = 0
Server.Port =
AmsServer.MaxThreads =

```

asdatabase.properties

The settings in the **asdatabase.properties** element are required in order to connect to the database used. The only database table used by the Archive Interface Server is the IMAGE_STATUS table. One exception is the AIS_PROTOCOL table is also used if 'UseProtocol' is activated.

```

JDBC.LoadDefaultDriver =
JDBC.DefaultConnection.Driver =
JDBC.DefaultConnection.DbURL =
JDBC.DefaultConnection.Catalog =
JDBC.DefaultConnection.DbUser =
JDBC.DefaultConnection.DbPassword =
UseSPQualifier =
Name.Size =
Timestamp.Size =

```

amsdatabase.properties

The settings in the **amsdatabase.properties** element are required in order to connect to the database used. The Archive Interface Server uses the IMAGE_STATUS and AIS_PROTOCOL (if activated) tables.

This property file is in most cases identical to the **asdatabase.properties** element described above.

```
JDBC.LoadDefaultDriver =
JDBC.DefaultConnection.Driver =
JDBC.DefaultConnection.DbURL =
JDBC.DefaultConnection.Catalog =
JDBC.DefaultConnection.DbUser =
JDBC.DefaultConnection.DbPassword =
UseSPQualifier =
Name.Size =
Timestamp.Size =
```

Status Entries

One of the following status settings is required for each of the table entry:

Process status settings:

Value	Process Status	Description
0	Registered	The item has been registered but the file (image or XML) has not yet been located. As soon as the item specified (IMAGE_LOCATION) has been located the status will be set to ready.
1	Ready	The item has been located and ready to be sent when requested.
2	Scheduled	This item has been scheduled to be sent in the reply for a request being processed currently. It will not be selected for any other requests. These items will be automatically reset to ready when the Archive Interface Server starts or manually. Should an error be determined by the Archive Interface Server they will be reset to ready (rollback).
3	Locked	This item was received by a client and no confirmation has been received.
4	OnHold	This item has been placed on hold by the client. As soon as a request has been made for this customer/account it will be released.
5	LockedForUser	This item has been locked for a specific user.
8	Rejected	Image has been processed and rejected by client. If DeleteCompletedFiles = 1 in archive.properties after the file has been deleted the status entry will be removed. If not set, the status entry must be manually removed after the file has been removed.
9	Done	Image has been processed and confirmed by client. If

Value	Process Status	Description
		<p><code>DeleteCompletedFiles = 1</code></p> <p>in archive.properties after the file has been deleted the status entry will be removed.</p> <p>If not set the status entry must be manually removed after the file has been removed.</p>

Error status settings:

Value	Error Status	Description
A	Lost	Image file could not be located.
B	Corrupt	Image file could not be processed.
C	Format	Invalid file format determined.
D	Delete error	Image file was specified to be deleted, delete could not be performed.
E	XML inprocessable	Invalid XML file.
F	Image inprocessable	The image file could not be processed.
G	XML Scan error	<p>The XML file contains invalid data.</p> <p>This occurs when scanning the XML file for BNO, CountryId, BankCode, CustomerNo / AccountNo.</p> <p>The error was located while scanning, not parsing. The XML file can not be further processed and is registered in the IMAGE_STATUS table.</p>
H	Image file name length	The name of the image file is too long to be registered.
I	XML File error from Client	The Client returned an error processing the XML file received from the AIS. The XML will not be deleted, the error code (and entry) remains in the AIS IMAGE_STATUS table.

Appendix

Property files overview

Business Model property files

The following contains a list of property files which are required in the Central Configuration for modules which include the Business Model.

Property File	Type	Origin
actions.properties	Common	spclient.jar de/softpro/signplus/client/resources
ARestrictions.properties	Common	spclient.jar de/softpro/signplus/client/resources
bnr.properties	Common	spclient.jar de/softpro/signplus/client/resources
CIInstructions.properties	Common	spclient.jar de/softpro/signplus/client/resources
comments.properties	Common	spclient.jar de/softpro/signplus/client/resources
comments_de.properties	Common	spclient.jar de/softpro/signplus/client/resources
comments_en.properties	Common	spclient.jar de/softpro/signplus/client/resources
comments_es.properties	Common	spclient.jar de/softpro/signplus/client/resources
currency.properties	Common	spclient.jar de/softpro/signplus/client/resources
database.properties	Common	spclient.jar de/softpro/signplus/client/resources
help.properties	Common	spclient.jar de/softpro/signplus/client/resources
help_de.properties	Common	spclient.jar de/softpro/signplus/client/resources
help_en_US.properties	Common	spclient.jar de/softpro/signplus/client/resources
help_es.properties	Common	spclient.jar de/softpro/signplus/client/resources
Instructions.properties	Common	spclient.jar de/softpro/signplus/client/resources

Property File	Type	Origin
kernel.properties	Common	spclient.jar de/softpro/signplus/client/resources
language.properties	Common	spclient.jar de/softpro/signplus/client/resources
language_de.properties	Common	spclient.jar de/softpro/signplus/client/resources
language_en.properties	Common	spclient.jar de/softpro/signplus/client/resources
language_en_GB.properties	Common	spclient.jar de/softpro/signplus/client/resources
language_es.properties	Common	spclient.jar de/softpro/signplus/client/resources
layout.properties	Common	spclient.jar de/softpro/signplus/client/resources
lists.properties	Common	spclient.jar de/softpro/signplus/client/resources
menu.properties	Common	spclient.jar de/softpro/signplus/client/resources
menuSignInInfo.properties	Common	spclient.jar de/softpro/signplus/client/resources
Scan.properties	Common	spclient.jar de/softpro/signplus/client/resources
SCClient.properties	Common	spclient.jar de/softpro/signplus/client/resources
SCClientmenu.properties	Common	spclient.jar de/softpro/signplus/client/resources
SCClientmenu_de.properties	Common	spclient.jar de/softpro/signplus/client/resources
server.properties	Common	spclient.jar de/softpro/signplus/client/resources
settings.properties	Common	spclient.jar de/softpro/signplus/client/resources
SignBase.properties	Common	spclient.jar de/softpro/signplus/client/resources
signdoc.properties	Common	spclient.jar de/softpro/signplus/client/resources

Property File	Type	Origin
sizes.properties	Common	spclient.jar de/softpro/signplus/client/resources
tablayout.properties	Common	spclient.jar de/softpro/signplus/client/resources
sp_utils.properties	Common	spclient.jar de/softpro/signplus/client/utlis
utlis_lang.properties	Common	spclient.jar de/softpro/signplus/client/utlis
utlis_lang_de.properties	Common	spclient.jar de/softpro/signplus/client/utlis
SVCmessages.properties	Common	softpro.jar ./
bno.properties*	Private	custom.jar de/demo/resources*
comments.properties*	Private	custom.jar de/demo/resources*
layout.properties*	Private	custom.jar de/demo/resources*
menu.properties*	Private	custom.jar de/demo/resources*
Scan.properties*	Private	custom.jar de/demo/resources*
SCClient.properties*	Private	custom.jar de/demo/resources*
SCClientmenu.properties*	Private	custom.jar de/demo/resources*
settings.properties*	Private	custom.jar de/demo/resources*
SignBase.properties*	Private	custom.jar de/demo/resources*
sizes.properties*	Private	custom.jar de/demo/resources*
tablayout.properties*	Private	custom.jar de/demo/resources*

“*” – Used for explanation purposes and will vary with each customer. The setting must be a valid input as required for the customer. The entries are included here only as sample info! The sample entries are from a FraudOne sample installation.

Thin Client property files

The following contains a list of property files which are required in the Central Configuration for the ‘InstanceType’ - ThinClient.

In addition the property elements for the Business Model (see section [Business Model Property Files](#)) are required.

Property File	Type	Origin
tcustom.properties	Private	SignPlus.war WEB-INF/classes
visibility.properties	Private	SignPlus.war WEB-INF/classes
custom.properties	Private	SignPlus.war WEB-INF/classes
tcversion.properties	Private	SignPlus.war WEB-INF/classes
version.properties	Common	SignPlus.war WEB-INF/lib/tclient.jar de/softpro/signplus/thinclient
ThinClient.properties	Common	SignPlus.war WEB-INF/lib/tclient.jar de/softpro/signplus/thinclient
ThinClient_de.properties	Common	SignPlus.war WEB-INF/lib/tclient.jar de/softpro/signplus/thinclient
ThinClient_en.properties	Common	SignPlus.war WEB-INF/lib/tclient.jar de/softpro/signplus/thinclient
ThinClient_es.properties	Common	SignPlus.war WEB-INF/lib/tclient.jar de/softpro/signplus/thinclient
ThinClient.properties	Private	SignPlus.war WEB-INF/lib/tclient.jar ./
ThinClient_de.properties	Private	SignPlus.war WEB-INF/lib/tclient.jar ./

Property File	Type	Origin
ThinClient_en.properties	Private	SignPlus.war WEB-INF/lib/tclient.jar ./
ThinClient_es.properties	Private	SignPlus.war WEB-INF/lib/tclient.jar ./

Archive Interface Server property files

The following contains a list of property files which are required in the Central Configuration for the Archive Interface Server ('InstanceType' – ArchiveServer).

In addition the property elements for the Business Model (see section [Business Model Property Files](#)) are required.

Property File	Type	Origin
archive.properties	Private	
ams.properties	Private	
asdatabase.properties	Private	
amsdatabase.properties	Private	
AlSmessages.properties	Private	archive.jar ./
SrvMngr4.ini	Common	[Common] section from SrvMngr4.ini
SrvMngr4.ini	Private	[ArchiveInterface] section from SrvMngr4.ini

Parameter overview

Server Manager parameters

The following table contains the description of the available Server Manager parameters.

Parameter	Description
4ASVSignatoryDisableNotCompare	If the signatory "not for ASV" and the parameter is set, the signatory isn't used for compares.

Parameter	Description
	If the signatory is for ASV and the parameter is set to No, the signatory will be used for verification. Default: No
Account	
ActiveDirectoryTechnicalUser	The technical user user-id to use to connect to Active Directory.
ActiveDirectoryTechnicalPassword	The technical user password (may be encrypted) to use to connect to Active Directory.
ActiveDirectoryPath	Additional path information to identify the Active Directory server which will process the Active Directory authentication and query requests. It is recommended not to set this parameter unless absolutely necessary.
ALDongle	Yes - if an Account Loader needs a dongle No - any Account Loader which requires a dongle will not start
ApplArgs	Any application parameter(s) required by the service program. Parameters are separated by blanks, a blank character in the parameter requires the whole parameter to be enclosed by quotes.
APSVBase	Forms the basis of the names of the system wide shared objects (shared memory, semaphores etc.) that are used by SignTeller Verification STV.
ARVQueue	Queue for ARV Default: 0
ARVResultFeatureID	The Feature ID of the result is written in the result table. Default: 0
AutomatQueueRefresh	Time (in seconds) until queue will be requested. Default: 30
AutomatQueues	Numbers of used queues Default: no default
AVC	AVC = Y Is needed ONLY when the Valued Customer automat does not run on the Application Server machine where the standard SignBase and SignCheck servers are running.
BankNo	
BNo	Optional: restrict the section to this BNo.

Parameter	Description
BNoListStringSize	Space to allocate to hold a list of BNOs in memory when configuring databases per BNO. The value must be large enough to hold all BNOs for the database with the largest list of BNOs. The default is 4096, large enough for the absolute maximum limit of 1000 BNOs.
BranchNo	Optional: restrict the section to this branch number (BNo must also be set).
ClassPath	Java classpath for FraudOne modules Example <code>.;service.zip;service.jar;softpro.jar;sputils.jar;tiff.zip;regex.jar;%CLASSPATH%;</code>
CommitCount	Items to process before doing a database commit.
ConfigFile	Name and path of WFRouter.ini file
CRSConfigFile	Name and path of the CRS configuration file
CRSDtdFile	DTD file that the rule definition will be verified against
CRSPort	Port used to receive CRS engines registration and messages
CRSRouterAddress	TCP/IP Address for CRS Router connection
CRSRouterPort	Port for CRS Router connection
CRSRuleFile	File or configuration element that contains the CRS rule definitions (needed to extract the queue list).
CSProtocol	Configuration server protocol entries required
CSSchema	Database schema name to use for the FraudOne Configuration database tables
CVBLVersion	Version number for Variant batch loader
Database	The SignBase database name. Name of the database with the SignPlus tables. Default: signplus
DatabaseCS	Configuration database name
DatabaseSC	The SignCheck database name. Database name for the SignCheck tables, if these are not contained in the database specified with 'Database='. Only for SignPlus E I.
Date	

Parameter	Description
DB2390	Must be set to 'Y', if SignBase data must be stored in a DB2/390 database.
DefaultCurrency	Currency code to use when an ICV or APSV document does not contain a currency. The default is "ZAR".
DelayOpenDB	Yes - DB logon after first request of the server
DeliverEmptyCheckStock	Enable / Disable check for Stockimages. Default: No
DocCheckWaitTime	The time (in seconds) that WFDecide waits, if it does not find any documents, before looking again.
DoNotAsk-IKnowWhatIAmDoing	
DtdFile	Name of the DTD file used to validate the rule definition.
DumpInboundMessage	Controls if a diagnostic trace is produced showing the binary contents of a server inbound message from the client. Default is "N" to suppress the diagnostic messages. Setting any other value for this parameter should be done only at the direction of technical support.
DynamicResolution	Selects the resolution in DPI of images produced by converting dynamic signatures. The default value is "0" which allows the images to take their own default resolution.
EIMigMode	Important This entry should not be changed without direction from Kofax.
Engine	Name of the engine DLL Default: SAE.dll
Extensions	Allocation of SignPlus extension tables. TEDCBA9876543210 -----Customer -----Account -----Signatory -----GroupIn -----Signature_M -----Signature_G -----Rule -----RuleGroup -----Signature_P -----Signature_C -----Signature_D -----Account_Image -----Document -----Mask -----Stock_Image -----Non-Customer Stock Image

Parameter	Description
	The extension tables allow the storage of data item attributes that are not part of the standard SignBase data model.
ForwardOnlyCursors	Controls the mode of operation of all cursors in ODBC access modules. Y - is the default but implies several restrictions on the operation of the SignBase server - in particular, search by name and search by personalid are not supported at this time. Default: Y
Group	Name of the server group
GroupAlias	Group alias name to use for the Attach Manager, Java service program or automat. Default is no group alias.
HeartBeat	After every n (n= 0-59) minutes the following heart beat display message is sent to the monitor log (n=0 no display): server's heart beat check %1 %2 %3 %4 %5 %6 %7 %8 message variables: %1: ,running: a %2: ,stopped: b %3: ,starting: d %4: ,stopping: e %5: ,crashed: f %6: ,waiting for work: g %7: ,error running: h %8: ,error stopped: i a-i: number of servers in the corresponding state. If a message variable contains no server, the variable will not be displayed.
HistoriseAImages	Y - means, that Account Images will be historized in any case for an update or delete N - this means Account Images are directly changed or deleted physically as it was so far Default: N
Hostname	
ICUAlias.%3.3u	An alternate three character BNO (usually alphanumeric) used by the customer in place of the standard three digit BNO. Example ICUAlias.001 = ABC
ICUConverter.%3.3u	The name of the ICU converter to be used to convert the national language data from the database into a form acceptable to downlevel client programs. The keys ICUConverter.000 through ICUConverter.999 are tested by the server and allow a different converter to be set for each BNO.

Parameter	Description
	<p>The value given for ICUConverter.999 is used as the default for BNOs that do not have their own specific value.</p> <p>This conversion is only relevant for down level clients (Releases earlier than 4.2).</p> <p>Example</p> <pre>ICUConverter.001 = Cp858</pre>
ICULocale.%3.3u	<p>The Locale to be used to convert searchable names to upper case. The keys ICULocale.000 through ICULocale.999 are tested by the server and allow a different locale to be set for each BNO. The value given for ICULocale.999 is used as the default for BNOs that do not have their own specific value.</p> <p>Example</p> <pre>ICULocale.001 = en_US</pre>
ICVDocPrefix	Document numbering prefix to ensure unique DOC_REF_NO for ICV generated SignCheck documents.
ICVGetterComment	A comment string to be placed into the result record generated for the ICV Getter step. Default is the Document Reference Number.
ICVGetterFeatureId	Feature identifier for ICV Getter results
ICVGetterQueue	ICV Getter queue
ICVLastFeatureId	Feature identifier of the document processing result that the ICV will use to assign an overall result to the document.
ICVPutterFeatureId	Feature identifier for ICV Putter results
ICVPutterQueue	ICV Putter queue
ICVPutterUser	User ID for ICV Putter results
ICVServerFactor	Scaling factor used in optimization of ICV server performance. A value between 1 and 10 controlling the number of simultaneous requests that a single ICV server can accept from the SDQ server.
ICVVisualFeatureId	Feature identifier used by the ICV visual queue simulation when it writes the results of its processing.
ICVVisualQueue	The queue on which the ICV internal workflow monitor will wait for documents to be processed as a visual verification step.
ICVVisualUser	The user name which will be used to lock documents and to identify results written for documents by the ICV visual queue simulation.

Parameter	Description
IdleTime	The time WFDecide waits without finding new documents before proceeding to the next section.
InsertHistorised	Important This entry should not be changed without direction from Kofax.
JavaDLL	The fully qualified file name of the DLL to use to create a Java Virtual Machine to start the Java program.
JavaOpts	Any JAVA start options like -verbose, -version, etc. With Sun Java, the memory must be limited, e.g. -Xmx200m
JCVersion	This parameter specifies a version number of a Java client. Any Java client with this version number will be logged with a DISPLAY message in the Monitor.
LastResult	0 - previous decision of this document is used as a result 1 - the result is taken from NewResult
Latin	List of signature kinds to be treated as latin: (1)Latin (2)Chinese (3)Japanese (4)Thai (5)Arabic (6)Russian (7)Greek (8)Hebrew
LicenceServerAddress	This parameter should be specified only when a licence server is to be used. A licence will be retrieved from the given TCP/IP address when it is required. If this parameter is not present, a local licence file will be used.
LicenceServerPort	The connection port on the licence server for the connection.
LicenceServerRetries	The number of times a failing connection to the licence server can be retried.
LicenceServerTimeout	The time (in milliseconds) to wait for a connection to be established with the license server.
LicenceUSBDriveLetter	This parameter may only be used when a USB Licence Device (hardware licence or SPLM2 dongle) has been provided. If this parameter is not present, a local licence file or licence transferred from a licence server will be used. When a hardware licence has been provided, the drive letter assigned to the USB device (the device must be inserted in a USB port to assign a drive letter) should be entered here. While this parameter is present, the USB licence device must remain inserted.

Parameter	Description
	Should the USB licence device be lost or damaged, the Server Manager must be stopped (SPService stop), this parameter removed and the Server Manager restarted to allow the SPLM2View utility to locate the locally installed licence files in order to obtain a licence installation code (or to resume using the locally installed licence files if these are not expired).
ListenBacklogLimit	The maximum number of connection requests that are allowed to be queued in the operating system waiting for the Attach Manager to accept them.
LiveInfo	Allows the Server Manager to start and monitor programs that do not themselves support the LiveInfo interface. Setting this parameter to 'Y' causes the Server Manager to start a proxy program which, in turn, starts the program to be monitored.
LoadHookDLL	Entry for loading hook functions in the customer specific ZU90H.DLL file. Y - ZU90H.DLL will be loaded N - ZU90H.DLL will not be loaded Default: N
Locale	
LockTimeout	Amount of time (in seconds) after which the workflow router releases blocked documents again.
LogEngineMatchRate	Yes - Additionally the original engine results will be written No - Only normalized result data will be output Default: No
LogFile	Path and name of the log file. The log file has to be stored on a local drive!
LogonPwd	Together with LogonUser, allows the Server Manager to start the monitored program(s) under a different user than which is running the Server Manager process itself. Currently unsupported, use only at the direction of technical support.
LogonUser	Together with LogonPwd, allows the Server Manager to start the monitored program(s) under a different user than which is running the Server Manager process itself. Currently unsupported, use only at the direction of technical support.
LoopDelayCycleTime	This entry should not be changed without direction from Kofax. It controls the communication between the application server and the Attach Manager.

Parameter	Description
MainClass	Java Class to be started. Example for DFP: <code>de.softpro.signplus.service.DFP</code>
MasterWait	This entry should not be changed without direction from technical support. It controls the queuing behaviour of the ICV and APSV servers and the associated SDQ instance.
MaxCrash	Controls the Server Manager restart behaviour, sets the number of times the Server Manager will attempt to restart a failed process.
MaxDocumentSize	The upper limit of the size of a front or back document image to be processed by ICV or APSV. The default is 1 MB. This controls the size of the shared memory used communicating between the SDQ instance and the APSV and/or ICV servers.
MaximumDocumentImages	The upper limit on the number of document images that will be returned when retrieving customer data. If a customer has more stored images than this limit, the query will return no binary image data and the client must read the binary image data for each image separately.
MaximumSearchResults	The maximum number of search results in a query.
MaximumStockImages	Maximum stock images displayed.
MaxSignatureSize	The upper limit of the size of a signature image to be processed by ICV or APSV. The default is 32 KB. This controls the size of the shared memory used communicating between the SDQ instance and the APSV and/or ICV servers.
MonLogPort	Port number for messages from the Server Monitor.
MonRequestPort	Port number for commands to the Server Monitor.
MsgTypeList	Message types that are supplied by the server. For the complete list refer to FraudOne Server Messages .
MsgVersionType	Compatibility parameter for the Clients: 1 - SignPlus E I 2 - SignPlus E II - Java-Client Not to change by the customer! Default: 2
Name	Heading in the title line of the cmd window.
NewConfigurationInterval	Controls the periodic reprocessing of the SrvMngr4.ini file by the Server Manager.

Parameter	Description
	Currently unsupported, use only at the direction of technical support.
NewConfigurationStartTime	Controls the periodic reprocessing of the SrvMngr4.ini file by the Server Manager. Currently unsupported, use only at the direction of technical support.
NewConfigurationStopTime	Controls the periodic reprocessing of the SrvMngr4.ini file by the Server Manager. Currently unsupported, use only at the direction of technical support.
NewConfigurationTime	Controls the periodic reprocessing of the SrvMngr4.ini file by the Server Manager. Currently unsupported, use only at the direction of technical support.
NewResult	Set this value as the new result (only if LastResult=0).
NoHistory	No - To be used by customers with historization Yes - To be used by customers without historization If this entry is missing, the automat probably will get no data from SignBase.
NoServer	Allows the configuration of a SignBase Attach Manager as a licence server (with no active SignBase servers associated with it).
NotifyTimeout	Maximum time (in milliseconds) to wait for result in ICV Putter queue
OnHold	Should this section process normal (0) documents, or documents that are on hold (1). If you want to do both, you'll need two sections.
Password	Entry for logon to the database. The maximum allowed length of a password is 16 characters. Can be set in comment when the system password corresponds to the SignPlus password.
PasswordChangeInterval	Time interval in days when a password needs to be changed
PasswordCS	Password for the Configuration database. The maximum allowed length of a password is 16 characters.
PasswordDuplicateCount	Number of allowed password duplicates.
PasswordDuplicateInterval	Number of days, within which a duplicate password is not allowed. Example: '180' means that a password may not be reused until approx. 6 months have passed since its last use.

Parameter	Description
PasswordRetryLimit	Number of times allows to retry to login
PasswordSC	Password for the SignCheck database. The maximum allowed length of a password is 16 characters.
Path	
PerformanceCriticalSignCheckLists	Enables some additional SignCheck document list types. Can be used only when the corresponding release key has been provided by technical support.
PipeTimeout	Maximum time (in seconds) that an answer to a client request sent to the server is expected by the attach manager. If the time expires and no answer was received from the server the attach manager records this incident in the log file and issues server busy to the client (as long as the client still waits for the answer).
Port	TCP port number for communication.
PriorityBoostRefresh	Time (in seconds) after which the router checks for documents needing a priority boost. Important This check takes some time. It is performed at router start and after PriorityBoostRefresh seconds.
PriorityBoostTimeout	Timeout (in seconds) after which documents that are not processed in a queue get a priority boost.
Program	Executable program file started by the Server Manager
PropagateExtension	This parameter allows multiple signatory update to work without changing the extensions of the multiple targets.
PropagateSignatory	This parameter allows multiple signatory update to work without changing the signatory data of the multiple targets.
PropagateSignature	This parameter allows multiple signatory update to work without changing the signature images of the multiple targets.
Queue	The queue number this section applies to.
RawReadSignatureM	Read all signature records; this is faster when many signatures are present and most are needed.
RemoteIni	Configuration file name for automats Default: automat2.ini
RepeatRequestTimeout	Number of minutes between licence checks
ReplyPipeSleep	This entry should not be changed without direction from Kofax. It controls the communication between the application server and the Attach Manager.

Parameter	Description
ReplyPipeWait	This entry should not be changed without direction from Kofax. It controls the communication between the application server and the Attach Manager.
RequestCycle	This entry should not be changed without direction from technical support. It controls the queuing behaviour of the ICV and APSV servers and the associated SDQ instance.
ResultCycle	This entry should not be changed without direction from technical support. It controls the queuing behaviour of the ICV and APSV servers and the associated SDQ instance.
RiskThreshold	Global risk threshold for decision evaluation. Used to compare against actual item risk and chose target types 'above' or 'below' threshold.
RouterHost	
RuleFile	Name of the CRS rule file to use (stored in the configuration server).
SanityCheckRefresh	Time (in seconds) after which the router does its sanity check - see if all components are still running.
SBBNOList.%d	The list of BNOs which will use this dynamic SignBase database.
SBDatabase.%d	Where <n> is replaced by a number from 1 to the SignBaseDatabaseCount value. The connection name of (one of) the dynamic SignBase database.
SBDynRef	No - This server cannot deskew dynamic reference signatures, which were captured via a pen pad. Yes - This server deskews dynamic reference signatures. If deskewing is configured, a license for the functions on dynamic signatures is needed to allow the loading of the corresponding module.
SBExtensions.%d	The list of extension tables activated for this dynamic SignBase database.
SBPassword.%d	The password to use when connecting to the dynamic SignBase database.
SBProtocol	This entry can switch off logging of SignBase actions. Important This entry should not be changed without direction from Kofax.

Parameter	Description
SBSchema	Database schema name to use for the SignBase customer database tables.
SBUserid.%d	The identifier of the user to connect to the dynamic SignBase database.
Sca2Ex	Is used in conjunction with parameter UseRestInst in the automat2.ini configuration file. If Sca2Ex = Y also rules that are restricted (via Extensions) could be considered as reason for reject. Default: N
SchedCache	Name of the XML format file used in the Clock Calendar feature
SchedDtdFile	Name of the DTD format file used in the Clock Calendar feature
SchedPort	Port for 24hour clock handler server
SchedServer	IP address or name of 24hour clock handler server
SchedXmlErrFile	If a schedule file contains errors, it will be renamed to this file and discarded.
SchedXmlFile	Schedule file to be used.
SchedXmlWorkFile	Working copy of the schedule file (will be created by the server)
SCSchema	Database schema name to use for the SignPlus Configuration database tables.
SCSnippetPrio	String of 3 digits maximum, controlling the search order of snippets: M - Monochrome P - Pseudocolor G - Grayscale No database access will be done for image types whose corresponding string is not specified within this entry. Default: MPG
SCSnippetPrioCheque	Defines the search priority for snippets for checks. This requires, that the field C_TXN in DEB.SC_INTERFACE has been filled with any data for checks. If this parameter is omitted, the value defined in SCSnippetPrio is used.
SCSnippetPrioGiro	Defines the search priority for giro forms. This definition is used if the field C_TXN is not set. If this parameter is omitted, the value defined in SCSnippetPrio is used.

Parameter	Description
ServerConcurrentStart	Allows the Server Manager to start or stop more than one monitored process at the same time.
ServerId	<p>Important for DB2/390! Only if DB2390 = Y</p> <p>The ServerId is necessary to generate a unique TIMESTAMP on the host. The ServerId must be defined differently for each active SB Server Process. It has to be a 2-digit number between 00 and 99.</p>
ServerManagerInternetNetworkPort	Communication port used by Server Manager processes to establish a network linking the processes together into a network.
ServerPipeRestartTime	<p>This parameter is for FraudOne system recovery purposes only for hanging servers or DB requests exceeding 30 minutes.</p> <p>Maximum time in minutes that the Attach Manager waits on a new client request on "OpenPipe" to server if the server is still busy after a pipe timed out. The Attach Manager kills the server, which in turn is being restarted by the SrvMngr4.</p> <p>Do not change this parameter, if not instructed to do so by Kofax support personnel.</p>
ServerPipeRestartTimeout	The maximum time the Server Manager allows for a process to resume correct operation when a request has timed out. This must be long enough for a request that is running longer than the PipeTimeout (and so causes a timeout report to the client) to complete.
ServerStartDelay	Servers will be started with 'ss' seconds delay after SrvMngr4 start. This may be useful in NT Service environment, to assure servers are started not until the database engine is fully up and running.
ServerTimeout	Maximum time (in seconds) that a request waits until a server is free. If after that time still all defined servers are busy, the attach manager returns to the client: server busy.
ServiceThreadStack	<p>This entry should not be changed without direction from Kofax.</p> <p>It controls the communication between the application server and the Attach Manager.</p>
SharedPath	<p>Fully qualified path to the SignPlus shared DLLs.</p> <p>Needed if SrvMngr4 is run as an NT service or as an NT cluster.</p>
ShowMode	<p>0 - no window on the user interface</p> <p>1 - icons on the user interface</p> <p>2 - extra window on the user interface</p>

Parameter	Description
SignBase	<p>N - No means that no read and write to SignBase tables is possible</p> <p>R - Read (no extensions) Only read access is possible using this server (loads Zu90R.dll)</p> <p>U - Update (no extensions) Read and update access is possible using this Server (loads Zu90W.dll)</p> <p>E - Extended Read (with extensions) Read-Access required for bank-specific data (loads Zu90R.dll and ZuREx90.dll)</p> <p>W - Write Extended (read & update with extensions) Write-Access required for bank-specific data (loads Zu90R.dll and ZuREx90.dll)</p>
SignBaseDatabaseCount	<p>The number of dynamic SignBase databases that are configured.</p> <p>Example</p> <pre>SignBaseDatabaseCount = 2 SBDatabase.1 = AX001 SBUserid.1 = signplus SBPassword.1 = sp01 SBBNOList.1 = 001,002,003 SBExtensions.1 = 012 SBDatabase.2 = AX002 SBUserid.2 = signplus SBPassword.2 = sp01 SBBNOList.2 = 100 SBExtensions.2 =</pre>
SignCheck	<p>Yes - This server provides Read and Update access to the SignCheck database</p> <p>No - This server provides no access to the SignCheck database</p>
SignSec	Y - This server provides User administration incl. Logon / Logoff
SignWareDLLName	<p>Name of SignWare DLL to be loaded. The default is the correct SignWare DLL for the current Attach Manager build.</p> <p>Note Use only under direction from Kofax support. Incorrect use may cause unrecoverable damage to signature data in the database.</p>
SkipInputSignatory	<p>This parameter allows multiple signatory update to work without updating any information associated with the supplied update target signatory. This is intended for the special situation where a client propagates the current contents of a specific signatory to all others in the multiple targets.</p>

Parameter	Description
SQLInterruptRetries	<p>Number of interrupt retries of running DB2 SQL Statements, before a rollback is performed.</p> <p>This entry is only valid with the combination of the SQLTimeOut setting.</p> <p>Default: 5</p>
SQLTimeOut	<p>Time (in seconds) after a long running DB2 SQL transaction of a SignBase or SignCheck server should be cancelled. An ODBC Server can only interrupt long running SignBase name search actions.</p> <p>Default: 0 (no timeout is set)</p>
Stage1Parm	Parameter used by the SignBase DFP to run the DFP scripts.
Stage1Path	Parameter used by the SignBase DFP to run the DFP scripts.
Stage1Pgm	Parameter used by the SignBase DFP to run the DFP scripts.
Stage2Parm	Parameter used by the SignBase DFP to run the DFP scripts.
Stage2Path	Parameter used by the SignBase DFP to run the DFP scripts.
Stage2Pgm	Parameter used by the SignBase DFP to run the DFP scripts.
StartTime	<p>The program is to be run between the time given in this StartTime parameter and the time given in the StopTime parameter. Between these times, the program will be started automatically if it is not running and outside these times it will be stopped automatically. If StartTime is after StopTime, the active time period extends over midnight to the StopTime on the next day.</p> <p>For StartTime to be effective, a StopTime other than 00:00 must be entered. Midnight can be entered as 24:00.</p> <p>Times should be entered in 24 hour format.</p> <p>The time parameters StartTime and StopTime are mutually exclusive with StartTrigger and StopTrigger. Specifying both results in undefined behaviour.</p> <p>Format: mm:hh</p> <p>Default: 00:00 (means start immediately)</p>
StartTimeoutFactor	Controls the time limit the Server Manager will wait when a server is started. This factor is used to multiple the PipeTimeout value to determine the startup time allowed.
StartTrigger	<p>If the time given in this StartTrigger parameter is reached, the program will be started if it is not running.</p> <p>Times should be entered in 24 hour format. (Midnight can be entered as 24:00)</p>

Parameter	Description
	This parameter (and StopTrigger) is mutually exclusive with StartTime and StopTime – specifying both results in undefined behaviour.
StatFile	Path and name of the statistics file. The log file has to be stored on a local drive!
Statistics	Yes - Database access counters and statistics will be recorded and written to the database No - Database access counters and statistics will not be recorded
StatisticsHold	Controls the frequency of database write operations when storing database statistics in the database. The number supplied here is the number of database access requests the server can process before the statistics are written to the database.
StopTime	See StartTime . Default: 00:00 – no stop Only valid if StartTime = 00:00
StopTrigger	If the time given in this StopTrigger parameter is reached, the program will be stopped if it is running. Times should be entered in 24 hour format. (Midnight can be entered as 24:00) This parameter (and StartTrigger) is mutually exclusive with StartTime and StopTime – specifying both results in undefined behaviour.
StopWhenReady	If true (!= 0) WFDecide stops after processing the last section. If not, it continues running and starts from scratch.
SuppressFlags	Prevent database access to known empty tables: Account table: 1 Signatory table: 2 Rule table: 4 RuleGroup table: 8 GroupIn table: 16 AccountImage table: 32 Document table: 64 Mask table: 128 StockImage table: 256 CustomerChangeUser: 512 Non-Customer Stock Image: 1024
TAPVersion	Version comparison encoding for a servlet API client.
TCP_SO_RCVBUF	Receive buffer size for TCP communication stack API.

Parameter	Description
TCP_SO_SNDBUF	Send buffer size for TCP communication stack API.
TcpConAttempts	Count of attempts to do a TCP/IP connection. Default: 3
TCPDebug	Enable verbose diagnostics for TCP communications. Causes severe performance degradation in the communication programs.
TCPLog	Enables writing the binary data received from a client to a diagnostic file. Causes severe performance degradation in the communication programs.
TCPLogFile	File to which the diagnostic information should be written
TcpMsgEncrypt	For specifically customized Java Clients only, that support TCP message encryption; Do not change this parameter in the supplied INI file.
TCPTimeout	The maximum time (in seconds) allowed for a client to complete a message transmission to the server (first byte to last byte).
TcpWaitTime	Waiting time (in milliseconds) between to do a TCP/IP connection Default: 1000
TCSVersion	Version comparison encoding for a Thin Client session
TCVersion	Version comparison encoding for a Thin Client
Timeout	
UnlockOnVisual	Controls the operation of the ICV Visual Queue simulation, automatically and immediately unlocks documents that appear in the queue being used to control the visual queue simulation.
UsePriority	Use the priority boost process: 1 - activated 0 - deactivated If the priority boost process is not used, item priority is not increased after a specific time.
UseProcessCredentials	Yes or No. If Yes, the authentication server should use the end user user-id and password sent with the transaction to connect to Active Directory in order to authenticate and authorize the user. If both this parameter and <code>UseTechnicalUserCredentials</code> are both No, the domain user under which the server is running is used as the connecting Active Directory User. This is the recommended setting.

Parameter	Description
User	
UseTechnicalUserCredentials	<p>Yes or No. If Yes, the authentication server should use the user-id and password configured in the INI file as a technical user to connect to Active Directory in order to authenticate and authorize the user.</p> <p>If both this parameter and <code>UseProcessCredentials</code> are both No, the domain user under which the server is running is used as the connecting Active Directory User. This is the recommended setting.</p>
Userid	<p>User ID for database connections</p> <p>Entry for logon to the database. Can be set in comment when the system user corresponds to the SignPlus user.</p>
UseridCS	User ID for the Configuration database
UseridDW	The database connection user-id needed to connect to the Data Warehouse. Not currently used.
UseridSC	User ID for the SignCheck database
VipMultiplier	<p>Multiplier used to increase the risk of items belonging to VIP accounts.</p> <p>1 - leaves the risk unchanged</p>
VirtualBufferLimit	<p>Defines the virtual buffer limits.</p> <p>Is the maximum size of the communication buffer. The value specified is in bytes and will be rounded up to the next multiple of 4096. If this value is exceeded while processing a client request, the server will report a critical error and terminate.</p> <p>Values in the range 4194304 (4M) to 134217728 (128M) are recommended for this parameter.</p> <p>Default: 67108864 (64M)</p>
VirtualBufferStart	<p>Defines the virtual buffer start.</p> <p>The initial size of the communication buffer. The value specified is in bytes and will be rounded up to the next multiple of 4096. If more than this amount is needed, the buffer size will be increased to the limit value specified in the VirtualBufferLimit parameter.</p> <p>Values in the range 4096 to 65536 are recommended for this parameter.</p> <p>Default: 8192 (8K)</p>
WaitLimit	<p>This entry should not be changed without direction from technical support.</p> <p>It controls the queuing behaviour of the ICV and APSV servers and the associated SDQ instance.</p>

Parameter	Description
WFCheckRights	Y - check if a user has worked on an item before. Note The workflow has to support this function (customer dependent). Decreases performance!
WFCheckUserInGetNext	Does WFGetNext have to check if the user has not processed the document returned in one of the previous queues? Possible values: 0, 1 Default: 0 = FALSE. Use with care, since WFGetNext performance decreases. The preferred way to do this is to disable users working on different levels via user permissions. Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.
WFGetNextServers	The list of servers to use for WFGetNext (if needed). Format is: <code>queue:servername:port,queue:servername:port, ...</code> If this is not set, classic WFGetNext with direct DB access is used. Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.
WFGNCursorTimeout	Time (in seconds) after which cursors in WFGetNext time out. The cursor is rebuilt after this time to insure priority processing. If no priority processing is desired this time can be set to a higher value. Default: 60 Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.
WFGNSErrorTimeout	If configured, WFGetNext will fall back to using direct DB access if the GetNext server can't be reached. This will continue for WFGNSErrorTimeout seconds, after which WFGetNext will try to reach the server again. The default is 0, which means that no fallback method is used. Can be set for all servers using the workflow API. Setting this item in the server section overrides the value in the common section.
WFPort	Port used to receive messages from other applications.
WFRouterAddr	
WFRouterPort	

Parameter	Description
WFServer	Name of the machine where the WFRouter is located. CRSSrv will connect to it and accept requests.
WFSocketTimeout	<p>If using GetNext servers, the timeout (in milliseconds) after which WGetNext gives up. GetNext will retry three times and resend the message another three times. Thus the total timeout will be 3 * 3 * WFSocketTimeout.</p> <p>Can be set for all FraudOne Workflow servers using the workflow API. Setting this item in the server section overrides the value in the common section.</p> <p>Default: 1000</p>
WFSrvErrTimeout	Time (in seconds) after which a connection is considered broken.
WFSrvPort	Port number WFSrv uses to accept connections (UDP).
WriteStatistics	<p>Yes - Store the statistics to database</p> <p>Default: Yes</p>
WriteTimeout	Maximum time (in seconds) that the attach manager waits from start sending the server's return data to the client (TCPWRITE) until the client accepts (reads) the data (close socket). If the client does not close the socket in time an error entry is put into the log.

Administration Client parameters

The following table contains the description of the available Administration Client parameters.

Parameter	Description
Blacklist.GlobalFraudSignatureList.Visible	
Blacklist.PayeeName.Visible	Visibility of blacklist module PayeeName
Blacklist.PayeeVerification.Visible	Visibility of blacklist module PayeeVerification
Blacklist.StockImage.Visible	Visibility of blacklist module Checkstock
Blacklist.Visible	Visibility of blacklist module Administration
CRS.DirectImport.Visible	<p>1 - means that it is possible to import a rule graph from XML-file directly into the system.</p> <p>This can only be used when the CRS Editor is not visible.</p> <p>Default: 0</p>

Parameter	Description
CRS.LoadActiveRule	Initial behaviour when starting the CRS Editor: 0 - No rule graph is loaded. 1 - The active rule graph is loaded. Default: 1
CRS.Visible	Visibility of module CRS Editor
CRS_DirectUpgrade.Visible	1 - means that the active rule graph can be upgraded to the latest CRS version. This can only be used when the CRS Editor is not visible.
CRS_Editor.RuleBlocks.Enabled	
HTMLBrowser.Path	Absolute path to the installed HTML browsers .exe file.
ParamAdmin.DisplayCRSEntries	1 - means that CRS elements are displayed in the System Configuration. These entries are coloured gray and read-only per default.
ParamAdmin.UseExtended	Switches between the extended and the simple GUI for System Configuration
ParamAdmin.Visible	Visibility of module System Configuration
UseDefaultBrowser	1 - means that the property HTMLBrowser.Path is ignored and the system's default web browser is used to display report data.
UserAdmin.QuickAccess.Visible	1 - means that the quick access panel is visible in the user administration. In this panel the user can quickly set access rights for a BNO.
UserAdmin.Visible	Visibility of module User Administration

Java Client parameters

The following table contains the description of the available Java Client parameters.

Parameter	Description
autoAssign	For all customer types will the variant be assigned automatically if there is only one signatory. Possible values: yes, no Default: no (no autoAssign)

Parameter	Description
<code>autoAssignCorporate</code>	For corporate customers will the variant be assigned automatically if there is only one signatory. Possible values: yes, no Default: The value of <code>autoAssign</code> .
<code>autoAssignOther</code>	For other customers will the variant be assigned automatically if there is only one signatory. Possible values: yes, no Default: The value of <code>autoAssign</code> .
<code>autoAssignPrivate</code>	For private customers will the variant be assigned automatically if there is only one signatory. Possible values: yes, no Default: The value of <code>autoAssign</code> .
<code>FocusColor</code>	Coloring of the current field with focus: black, blue, cyan, dark grey, grey, light grey green, magenta, orange, pink, white, yellow
<code>HideUnboundVariants</code>	Since release 3.5.45 1 - Unbound variants are not displayed 0 - Display unbound variants Default: 0
<code>ImageStack.Root.Path</code>	The root directory of the image loader. Example <code>C:/temp/test/Stapel/</code>
<code>ImageStack.Suffix</code>	Valid image file suffixes for the image loader. Multiple entries are separated by empty space. Example <code>bmp gif jpg jpeg tiff</code>
<code>maxVariants</code>	The maximum number of allowed variants for all customers Default: 0
<code>maxVariantsAutoAssign</code>	The maximum number of allowed variants for all customers if <code>autoAssign = yes</code> Default: The value of <code>maxVariants</code> .
<code>maxVariantsAutoAssignCorporate</code>	The maximum number of allowed variants for corporate customers if <code>autoAssign = yes</code> Default: The value of <code>maxVariantsAutoAssign</code> .

Parameter	Description
maxVariantsAutoAssignOther	The maximum number of allowed variants for other customers if autoAssign = yes Default: The value of maxVariantsAutoAssign.
maxVariantsAutoAssignPrivate	The maximum number of allowed variants for private customers if autoAssign = yes Default: The value of maxVariantsAutoAssign.
maxVariantsCorporate	The maximum number of allowed variants for corporate customers Default: The value of maxVariants.
maxVariantsOther	The maximum number of allowed variants for other customers Default: The value of maxVariants.
maxVariantsPrivate	The maximum number of allowed variants for private customers Default: The value of maxVariants.
Overify	Verification rate for other customers
Pverify	Verification rate for private customers
QueueEmulation.ICFileName	File name to load that contains the data to be processed
QueueEmulation.ICFilePath	Path where to find the Inwards Clearing file
ScanCtrl.BWBitmapSizeInt	Max Size of the edited BlackWhite-Image. Exceeding this size will result in a PopupMessage: please reduce the selected size.
ScanCtrl.CheckSizeInt	0 - Do not check sizes 1 - Check Size and popup a warning 2 - Check size, popup a warning and return to scan Dialog, if size too large 3 - Check size, popup a warning and return to edit Dialog, if size too large Default: 1
ScanCtrl.EditResult	1 - Erase the edited Region in the ScanImage 2 - Subtract the edited Region from the ScanImage 3 - Copy the edited Region into the ScanImage 4 - Display ScanImage unchanged 5 - Display a Rectangle in the ScanImage 6 - Mask the ScanImage with the edited Bitmap (Bitwise: AND)
ScanCtrl.GrayBitmapSizeInt	Max Size of the edited Gray-Image.

Parameter	Description
	Exceeding this size will result in a PopupMessage: please reduce the selected size
ScanCtrl.PadDriver	-1 - Search Drivers, use the first one found 0 - No Drivers are installed 1 - Use Wacom Driver 2 - Use MobiNetix driver
ScanCtrl.PadHeight	Pad-Height (in mm)
ScanCtrl.PadUseLCD.bool	0 - An LCD pad will not be used 1 - An LCD pad will be used (e.g. Wacom PL-400), where the sensitive area corresponds to the display area. In this case the coordinates of the pencil have to be mapped with coordinates of the display area. Default: 0
ScanCtrl.PadWidth	Pad-Width (in mm)
ScanCtrl.ResolutionErrAction	Action if resolution of imported image is less than 200 dpi. 1 - Query action 2 - Accept all 3 - Accept none 4 - Warn and accept none 5 - Warn and accept all 0 - If Proper available warn and accept none, else query action
ScanCtrl.TwainDriversInstalled	0 - No Twain Driver is installed 1 - At least one Twain Driver should be installed (otherwise the client stops at once) 2 - Search Twain Drivers, use the first one found
ShowVerifyIcon	Rolling eyes displayed in status line when customer and/or account have not yet been released.
UIVerify	Verification rate for corporate customers

Thin Client parameters

The following table contains the description of the available Thin Client parameters.

Parameter	Description
ActiveXVersion = 1,6,2,1	The ActiveXVersion parameter defines the minimum version of the ActiveX object which needs to be installed on the client (MS Internet Explorer) in order to run the Thin Client.
BankApplicationName	Name to be used as display name for standard Thin Client application. It is used in both Teller and Thin Client operation modes. This name appears in the page title and in the logo header. Default: "SignPlus for Demo Bank"
BankSiceApplicationName	Name to be used as display name for SICE client application. It is used in both Teller and Thin Client operation modes. This name appears in the page title and in the logo header. Default: "SignInfo Classic Edition for Demo Bank"
BrowserShortcut	See DisableBrowserShortcuts . Each function key that should be disabled can be configured with +, -, F1-F12, A-Z, 0-9, INS and one of the additional keys SHIFT, CTRL and ALT. A Thin Client function key definition can be one key or a combination of 2 keys. For a combination of two keys to have to start with SHIFT, CTRL or ALT followed by a separator (#) and the second key. Example BrowserShortcut = CTRL#P F1 SHIFT#F2 ALT#+
ChangePwd	Specifies whether or not password changing is enabled. If set to 1 the "Change password" menu entry will be available and this functionality will be enabled. Setting the property to 0 will disable password changing for all users. This setting is only considered if UseSignSec = 1 is set in the custom.properties file. Default: 1
CheckStockResultList.MaxHeight	CheckStock Result List Default: 200
CheckStockResultList.MaxWidth	CheckStock Result List Default: 400
CheckStockResultList.ScrollingStrategy	CheckStock Result List Default: 1
CheckStockResultListPreview.MaxHeight	CheckStock infotip in Result List

Parameter	Description
	Default: 300
CheckStockResultListPreview.MaxWidth	CheckStock infotip in Result List Default: 700
CheckStockResultListPreview.ScalingStrategy	CheckStock infotip in Result List Default: 1
CheckStockShowDetailsOnRight	<p>This property applies to the stock images displayed in the SignBase application mode, and to the stock images displayed as reference data in SignCheck.</p> <p>The property specifies whether stock image details should be shown on the right of the image instead of below the image. This is useful in situations where the screen resolution is small.</p> <p>Setting the value to true will result in image details being displayed to the right of the image.</p> <p>Setting the value to false will result in the image details being displayed below the image.</p> <p>Default: true</p>
CloseTellerPopupOpener	<p>Determines whether or not the opener window of the Teller popup request will be closed. Only valid for Ups2Teller Interface.</p> <p>Default: true</p>
ComplexRulesView.MaxHeight	<p>Signature in rules display</p> <p>Default: 300</p>
ComplexRulesView.MaxWidth	<p>Signature in rules display</p> <p>Default: 300</p>
ComplexRulesView.ScalingStrategy	<p>Signature in rules display</p> <p>Default: 0</p>
DefaultShowSignatory	<p>Specify which SignBase view should be displayed by default.</p> <p>After enabling this option, the list of all signatories belonging to the loaded customer / account is displayed by default as a search result in SignBase.</p> <p>Default: 1</p>
DefaultShowSignInfo	<p>Specify which SignBase view should be displayed by default.</p> <p>Default: 1</p> <p>Note The SignInfo setting only takes effect if SignInfo is enabled via custom.properties and SignInfo information is only displayed if the account/customer contains any image data.</p>
DefaultShowStockImage	Specify which SignBase view should be displayed by default.

Parameter	Description
	<p>Default: 1</p> <p>Note The checkstock setting only takes effect if Checkstock is enabled via custom.properties and that checkstock results are only displayed if the account/customer contains any image data.</p>
<code>DisableAllMenuEntries</code>	<p>The setting true disables all menu entries and overrides any other settings.</p> <p>Default: false</p> <p>This property can be differentiated in addition with values true or false for the following property names.</p> <ul style="list-style-type: none"> All teller menus can be disabled. Overrides <code>DisableAllMenuEntries</code>: <code>DisableAllMenuEntries.Teller</code> Only menus in Kld2Teller mode can be disabled. Overrides <code>DisableAllMenuEntries</code>: <code>DisableAllMenuEntries.Teller.Kld2Teller</code> Only menus in Ups2Teller mode can be disabled. Overrides <code>DisableAllMenuEntries</code>: <code>DisableAllMenuEntries.Teller.Ups2Teller</code>
<code>DisableAllMenuEntries.Teller</code>	<p>All teller menus can be disabled. See also DisableAllMenuEntries.</p> <p>Overrides <code>DisableAllMenuEntries</code>.</p>
<code>DisableAllMenuEntries.Teller.Kld2Teller</code>	<p>Only menus in Kld2Teller mode can be disabled. See also DisableAllMenuEntries.</p> <p>Overrides <code>DisableAllMenuEntries</code>.</p>
<code>DisableAllMenuEntries.Teller.Ups2Teller</code>	<p>Only menus in Ups2Teller mode can be disabled. See also DisableAllMenuEntries.</p> <p>Overrides <code>DisableAllMenuEntries</code>.</p>
<code>DisableBrowserShortcuts</code>	<p>Specifies whether or not shortcuts (browser shortcuts and Thin Client shortcuts) should be disabled.</p> <p>All shortcuts which should be disabled need to be listed and separated by blanks as values for additional property entry <code>BrowserShortcuts</code>.</p> <p>Default: false</p>
<code>DisableBrowserToolbars</code>	<p>Specifies whether or not the Thin Client should be opened in a new popup window without toolbar, menubar, statusbar or addressbar.</p> <p>Default: false</p>
<code>DisableDrag</code>	<p>Specifies whether drag-able page objects should be disabled. If set to false, drag-able objects may be moved (via keep pressing</p>

Parameter	Description
	<p>the left mouse button) around the page and dropped (e.g. function icons). Signatures can be moved also (e.g. for a visual comparison), but after releasing the left mouse button the signature snaps back to the original position.</p> <p>Default: false</p>
<code>DisableHelpButton</code>	<p>Specifies whether or not the Help button should be disabled/hidden. Set to true to disable.</p> <p>This property can be differentiated in addition with values true or false for the following property names.</p> <ul style="list-style-type: none"> • The Help button in teller mode can be disabled. Overrides <code>DisableHelpButton</code>: <code>DisableHelpButton.Teller</code> • Enables or disables the Help button in Kld2Teller mode. Overrides <code>DisableHelpButton</code>: <code>DisableHelpButton.Teller.Kld2Teller</code> • Enables or disables the Help button in Ups2Teller mode. Overrides <code>DisableHelpButton</code>: <code>DisableHelpButton.Teller.Ups2Teller</code>
<code>DisableHelpButton.Teller</code>	<p>The help button in teller mode can be disabled. See also DisableHelpButton.</p> <p>Overrides <code>DisableHelpButton</code>.</p>
<code>DisableHelpButton.Teller.Kld2Teller</code>	<p>Enables or disables the help button in Kld2Teller mode. See also DisableHelpButton.</p> <p>Overrides <code>DisableHelpButton</code>.</p>
<code>DisableHelpButton.Teller.Ups2Teller</code>	<p>Enables or disables the help button in Ups2Teller mode. See also DisableHelpButton.</p> <p>Overrides <code>DisableHelpButton</code>.</p>
<code>DisableImageTools</code>	<p>Disables the image tools in SignBase Clients i.e. Image transformation tools such as rotate, clean, invert in SignBase Clients.</p> <p>Default value: false</p>
<code>DisableRightMouseClicked</code>	<p>Specifies whether or not the right mouse click should be disabled within all pages of the application.</p> <p>Default: false</p>
<code>DisableToolTips</code>	<p>Specifies whether or not link tool tips or info tips should be disabled. This includes any image previews, e.g. SignInfo preview, or SignWith short lists.</p> <p>Default: false</p>

Parameter	Description
<code>displayLogo</code>	<p>Specifies whether or not the SignPlus logo header should be displayed on all pages.</p> <p>Set this property to 0 to remove the header.</p> <p>Default: 1</p> <p>The display of the logo can be differentiated in addition with values 0 or 1 for the following property names.</p> <ul style="list-style-type: none"> All teller modes will be displayed with or without logo. Overrides <code>displayLogo</code>: <code>displayLogo.Teller</code> Only Kld2Teller mode is displayed with or without logo. Overrides <code>displayLogo.Teller</code>: <code>displayLogo.Teller.Kld2Teller</code> Only Ups2Teller mode is displayed with or without logo. Overrides <code>displayLogo.Teller</code>: <code>displayLogo.Teller.Ups2Teller</code>
<code>displayLogo.Kld2Teller</code>	<p>Only Kld2Teller mode is displayed with or without logo. See also displayLogo.</p> <p>Overrides <code>displayLogo.Teller</code>.</p>
<code>displayLogo.Teller</code>	<p>All teller modes will be displayed with or without logo. See also displayLogo.</p> <p>Overrides <code>displayLogo</code>.</p>
<code>displayLogo.Ups2Teller</code>	<p>Only Ups2Teller mode is displayed with or without logo. See also displayLogo.</p> <p>Overrides <code>displayLogo.Teller</code>.</p>
<code>EnableAboutMenu</code>	<p>Specifies whether or not the About option should be displayed in the menu bar. Set to false to disable.</p> <p>Default: true</p> <p>This property can be differentiated in addition with values true or false for the following property names:</p> <ul style="list-style-type: none"> About menu in Teller mode can be disabled. Overrides <code>EnableAboutMenu</code>: <code>EnableAboutMenu.Teller</code> Enables or disables About menu in Kld2Teller mode. Overrides <code>EnableAboutMenu</code>: <code>EnableAboutMenu.Teller.Kld2Teller</code> Enables or disables About menu in Ups2Teller mode. Overrides <code>EnableAboutMenu</code>: <code>EnableAboutMenu.Teller.Ups2Teller</code>

Parameter	Description
<code>EnableAboutMenu.Teller</code>	Enables or disables About menu in Teller mode. See also EnableAboutMenu . Overrides <code>EnableAboutMenu</code> .
<code>EnableAboutMenu.Teller.Kld2Teller</code>	Enables or disables About menu in Kld2Teller mode. See also EnableAboutMenu . Overrides <code>EnableAboutMenu</code> .
<code>EnableAboutMenu.Teller.Ups2Teller</code>	Enables or disables About menu in Ups2Teller mode. See also EnableAboutMenu . Overrides <code>EnableAboutMenu</code> .
<code>EnableFontSelect</code>	Enables whether the user should be able to select the font size within the application. Default: true
<code>EnableFullDocumentHistory</code>	If set to false only the history of the workflow is displayed, otherwise the full history is displayed (including APIA-results). Default: true
<code>EnableLogoffMenu</code>	Specifies whether or not the Logoff option should be displayed in the menu bar. Set to false to disable. Default value: true
<code>EnablePhysicalVerification</code>	Set to true if physical verification (inwards-clearing) should be enabled. Default: false
<code>EnablePrimanota</code>	Set to true if primanota status should be available in SignCheck. Default: true
<code>EnableQueueRightsForStatus</code>	Set to true if SignCheck status should only be displayed for queues the user has rights for. Set to false if SignCheck status should be displayed for all queues (independent of queue rights). Default: true
<code>EnableSignatureCapture</code>	Set to true if dynamic signature capturing via pad should be enabled. Default: false
<code>EnableSignatureReplace</code>	Set to true if an existing signature can be replaced by capturing a new signature via pad. Default: false
<code>EnableSignCheckHistory</code>	Specify whether or not the SignCheck document history should be displayed by default when a document is loaded.

Parameter	Description				
	Default value: false				
EnableSignCheckLists	Set to true if SignCheck list processing should be enabled. Default value: true				
EnableSignCheckQueues	Set to true if queue processing should be enabled in SignCheck clients. Default value: true				
EnableSignCheckStatus	Set to true if status should be available in SignCheck. Default value: true				
EnableSignCheckVBD	Set to true if the VBD (Visual Branch Display) view should be visible as a menu item in SignCheck clients. Default value: false				
EnableSignCheckViewPersistence	Specify whether or not SignCheck views (i.e. the currently displayed data such as front, signature, reference data etc.) should remain persistent for the rest of the session when a user has changed them. If set to true, the views will remain persistent. If set to false, the view will be reset to the SignCheck configuration for every newly loaded document. Default: true				
EnableStoreAsGlobalFraud	Allows signatures from a SignCheck document snippet to be stored in SignBase as a global fraud in the blacklist. Default: true				
EnableStoreAsVariant	Set to true if storing of variants should be available in SignCheck. Default value: true				
EnableTellerErrorDialog	Determines whether or not error dialog should be displayed after failed teller request. The error is reported to the calling application (ReturnTo parameter) if set to false. Default: false				
EnableTellerShortcuts	Specifies whether or not shortcuts are enabled in the Teller operation mode. When set to true, the shortcuts can be configured according to the following property names: <table border="1"> <thead> <tr> <th>Property Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>ShortcutTellerAccept</td><td>Back to Teller and accept. Default: SHIFT#+</td></tr> </tbody> </table>	Property Name	Description	ShortcutTellerAccept	Back to Teller and accept. Default: SHIFT#+
Property Name	Description				
ShortcutTellerAccept	Back to Teller and accept. Default: SHIFT#+				

Parameter	Description																												
	Property Name	Description																											
	ShortcutTellerReject	Back to Teller and reject. Default: SHIFT#-																											
	Each shortcut can be configured with +, -, F1-F12, A-Z, 0-9, INS and additional with SHIFT, CTRL, ALT.																												
EnableThinClientShortcuts	<p>Specifies whether or not shortcuts are enabled in the standard Thin Client operation mode. When set to true, the following shortcut keys are enabled in the browser and can be changed.</p> <p>Default: true</p> <p>Each shortcut can be configured with +, -, F1-F12, A-Z, 0-9 and additional with SHIFT, CTRL, ALT (see also BrowserShortcut).</p> <table> <tr> <th>Property Name</th><th>Shortcut Key (default)</th><th>Functionality</th></tr> <tr> <td>ShortcutSearchAccount</td><td>SHIFT-F3</td><td>Search account</td></tr> <tr> <td>ShortcutSearchCustomer</td><td>F3</td><td>Search customer</td></tr> <tr> <td>ShortcutOpenHelp</td><td>F1</td><td>Open help</td></tr> <tr> <td>ShortcutAcceptGetNext</td><td>F12</td><td>Accept/Get Next (in SignCheck views)</td></tr> <tr> <td>ShortcutRejectGetNext</td><td>F9</td><td>Reject/Get Next (in SignCheck views)</td></tr> <tr> <td>ShortcutHoldGetNext</td><td>F10</td><td>Hold/Get Next (in SignCheck views)</td></tr> <tr> <td>ShortcutGetNext</td><td>F5</td><td>Get Next (in SignCheck views)</td></tr> <tr> <td>ShortcutDocumentHistory</td><td>F4</td><td>Check Decisions i.e. document history (in SignCheck views)</td></tr> </table>		Property Name	Shortcut Key (default)	Functionality	ShortcutSearchAccount	SHIFT-F3	Search account	ShortcutSearchCustomer	F3	Search customer	ShortcutOpenHelp	F1	Open help	ShortcutAcceptGetNext	F12	Accept/Get Next (in SignCheck views)	ShortcutRejectGetNext	F9	Reject/Get Next (in SignCheck views)	ShortcutHoldGetNext	F10	Hold/Get Next (in SignCheck views)	ShortcutGetNext	F5	Get Next (in SignCheck views)	ShortcutDocumentHistory	F4	Check Decisions i.e. document history (in SignCheck views)
Property Name	Shortcut Key (default)	Functionality																											
ShortcutSearchAccount	SHIFT-F3	Search account																											
ShortcutSearchCustomer	F3	Search customer																											
ShortcutOpenHelp	F1	Open help																											
ShortcutAcceptGetNext	F12	Accept/Get Next (in SignCheck views)																											
ShortcutRejectGetNext	F9	Reject/Get Next (in SignCheck views)																											
ShortcutHoldGetNext	F10	Hold/Get Next (in SignCheck views)																											
ShortcutGetNext	F5	Get Next (in SignCheck views)																											
ShortcutDocumentHistory	F4	Check Decisions i.e. document history (in SignCheck views)																											
EnableWatermark	<p>This parameter applies to the signatures, account images and check stock images. When enabled a watermark will be placed on the images shown in SignBase and the SignBase view in SignCheck. The Watermark is always a text entry and can be configured as described in the following properties.</p> <p>Default: false</p>																												
GetSpecificLock	<p>Do not lock documents when selected. Also avoids getNext when the document is not in a visual queue. Recommended when using the SP Teller Interface to verify checks.</p> <p>Default: true</p>																												

Parameter	Description
HandleSessionTimeout	Specifies whether or not the user should automatically be redirected to the login page when a session timeout occurs. Default: true Note This feature is disabled (false) in Teller operation mode.
help.de.sice	Relative path to the start page of the Thin Client help for SICE application mode (German language). Default: help/de/sice/default.htm
help.de.thin	Relative path to the start page of the Thin Client help for standard application mode (German language). Default: help/de/thin/default.htm
help.en.sice	Relative path to the start page of the Thin Client help for SICE application mode (English language). Default: help/en/sice/default.htm
help.en.thin	Relative path to the start page of the Thin Client help for standard application mode (English language). Default: help/en/thin/default.htm
ImageSmoothing	This property specifies whether or not images are smoothed before being sent from the server to the browser. Setting this property to 1 will result in better quality images but will slow down image transfer and processing. Setting this property to 0 will result in better performance, but poorer quality images. Default: 1
ImageView.MaxHeight	Signature in zoom view Default: 400
ImageView.MaxWidth	Signature in zoom view Default: 500
ImageView.ScalingStrategy	Signature in zoom view Default: 0
IsTeller	This property controls whether or not the application will operate in the Teller operation mode. Set this property to true if the context should be set up for Teller operation. Default: false
JDBCClass.DB2UDB	Class definition for DB2UDB JDBC access. Default entry in tcustom.properties file: com.ibm.db2.jcc.DB2Driver

Parameter	Description
JDBCClass.DB2ZOS	Class definition for DB2ZOS JDBC access. Default entry in tcustom.properties file: com.ibm.db2.jcc.DB2Driver
JDBCClass.MSSQL	Class definition for MSSQL JDBC access. Default entry in tcustom.properties file: com.microsoft.jdbc.sqlserver.SQLServerDriver
JDBCClass.ORACLE-OCI	Class definition for ORACLE-OCI access. Default entry in tcustom.properties file: oracle.jdbc.OracleDriver
JDBCClass.ORACLE-THIN	Class definition for ORACLE-THIN access. Default entry in tcustom.properties file: oracle.jdbc.OracleDriver
JDBCURL.DB2UDB	JDBC URL for DB2UDB access. Default entry in tcustom.properties file: jdbc:db2:%(d)
JDBCURL.DB2ZOS	JDBC URL for DB2ZOS access. Default entry in tcustom.properties file: jdbc:db2:%(d)
JDBCURL.MSSQL	JDBC URL for MSSQL access. Default entry in tcustom.properties file: jdbc:microsoft:sqlserver://%(s);DatabaseName=% (d);SelectMethod=Cursor
JDBCURL.ORACLE-OCI	JDBC URL for ORACLE-THIN access, default entry in tcustom.properties file: jdbc:oracle:oci:@%(d)
JDBCURL.ORACLE-THIN	JDBC URL for ORACLE-THIN access. Default entry in tcustom.properties file: jdbc:oracle:thin:@% (s):1521:%(d)
jpegQuality	JPEG Compression factor in the range from 0.0 to 1.0: 0.0 - lowest Quality 1.0 - highest Quality Default: 0.4
Kld2TellerStyle	CSS style sheet to be applied to all pages in Kld2Teller servlet mode. Default: /jsp/styles/SignPlus-styles-xs.css or if commented out the setting of Default Style
LDAP.Context	Set LDAP context class which is used. Default: com.sun.jndi.ldap.LdapCtxFactory

Parameter	Description
LDAP.DN	Distinguished name (DN) which describes the attributes in the LDAP tree down to the searched objectclass for the user information. host=tsys,o=yourbank,c=us,cn=localhost No default value set.
LDAP.Host	Set IP address for the LDAP server to connect to. Default: 127.0.0.1
LDAP.Objectclass	Objectclass name under which the user information is stored in the LDAP tree. No default value set.
LDAP.Port	Set port to connect to the LDAP server. Standard port for LDAP connections is 389. Default: 389
LDAP.Version	Set LDAP protocol version. Depends on which protocol the LDAP server can handle. Usually version 2 or version 3. Default: 30
LogFile	Trace output file (e.g. c:\tclient.log). Default: console output See also chapter Static Logging setup .
Login.BankCode	
Login.Bno	
Login.BranchCode	
Login.Cid	
LogLevel	Trace log level, can have one of the following settings: off, fine, info, warning, severe, all Default: severe See also chapter Static Logging setup .
LogoImageName	Specifies the image name/path which is used as logo on top of the Thin Client pages. Since this image location path is appended to the current path tomcat\webapps\SignPlus it must start with a slash. Default: /res/logo2.gif
NewWindowHeight	Defines the window size (height) if the Thin Client is opened in a new popup window (DisableBrowserToolbars = true).

Parameter	Description
	Default: 600 (pixels)
NewWindowWidth	Defines the window size (width) if the Thin Client is opened in a new popup window (DisableBrowserToolbars = true). Default: 800 (pixels)
PadBackImage1	A dynamic signature capturing compromises signing 3 times per pad. For each step a different pad background image can be defined. This settings specifies a generic first background image, independent from the pad LCD resolution). Default: PadBack1.bmp
PadBackImage1.widthxheight	Defines the specific background images for dynamic signature capturing, dependent from the pad LCD resolution. If you have attached a pad with LCD resolution 320 pixel (width) on 240 pixel (height), you should define images also with this resolution. Example <code>PadBackImage1.320x240 = PadBack1_320x240.bmp</code> Default: PadBackImage1.bmp
PadBackImage2	Specifies a generic second background image for dynamic signature capturing, independent from the pad LCD resolution). Default: PadBack2.bmp
PadBackImage2.widthxheight	See PadBackImage1.widthxheight Default: PadBackImage2.bmp
PadBackImage3	Specifies a generic third background image for dynamic signature capturing, independent from the pad LCD resolution). Default: PadBack3.bmp
PadBackImage3.widthxheight	Specifies a generic third background image for dynamic signature capturing, independent from the pad LCD resolution). Default: PadBackImage3.bmp
PluginVersion.Major = 1 PluginVersion.Minor = 13 PluginVersion.Revision = 2 PluginVersion.Build = 1	The four PluginVersion.xxx parameter define the minimum version of the plugin object which needs to be installed on the client (NS, Mozilla, Firefox) in order to run the Thin Client.
RulesTextSignatoryListClipLength	Sets the maximum length of the rules text in the signatory list in which they appear before they are clipped. The clipped portion of the text will then be replaced by "...". The Length is specified in number of characters. Default value: default value (set by cutoff attribute for SignatoryTag property singleRule in sigowentry-rules.jsp) is 0
RulesTextSignatoryListClipLength.Teller	Same as RulesTextSignatoryListClipLength , but only valid for teller modes.

Parameter	Description
	Overrides RulesTextSignatoryListClipLength
RulesTextSignatoryListClipLength.Teller.Kld2Teller	Same as RulesTextSignatoryListClipLength , but only valid for Kld2Teller servlet interface. Overrides RulesTextSignatoryListClipLength.Teller.
RulesTextSignatoryListClipLength.Teller.Ups2Teller	Same as RulesTextSignatoryListClipLength , but only valid for Ups2Teller servlet interface. Overrides RulesTextSignatoryListClipLength.Teller.
ScaleToGray	ICV display setting: If set to true, black and white images will be displayed in grayscale when the value of Zoom is < 100.0. This improves the quality of the displayed image which would otherwise appear to break up at low zoom factors. The underlying image held by csXImage (name of twain utility plugin) is not affected, only the view of it that is displayed. Default: true
SCArchive.Connect.DBMS	Defines the used database management system. Possible values: DB2UDB, DB2ZOS, ORACLE-THIN, ORACLE-OCI, MSSQL Default: DB2UDB
SCArchive.Connect.DBName	Name of the database. Default: signplus
SCArchive.Connect.DBServer	DB server name (necessary for MS SQL Server and for ORACLE ThinDriver) No default
SCArchive.Connect.PW	Password for database access. No default
SCArchive.Connect.Schema	DB schema name (qualifier). Default: dwsp
SCArchive.Connect.User	Userid for database access. No default
SCArchive.Image.Enabled	Specifies whether short term archive is checked for check images.
SCArchive.Image.Impl	Class definition for archive image access. Default: de.softpro.signplus.thinclient.signcheck.SCArchive
SCArchive.Resultlist.MaxRows	Max number of result rows for archive search. Default: 200

Parameter	Description															
SetTwainAutoBright	<p>ICV Twain setting for scanner.</p> <p>Setting this property to true will enable the Automatic Brightness function of the device, if available.</p> <p>Note This property only has any effect with devices that support such a feature.</p> <p>Default: true</p>															
SetTwainAutoDeskew	<p>ICV Twain setting for scanner.</p> <p>Setting this property to true will enable the Automatic Deskew Correction feature of the device, if available. This will rotate the image received from the device (usually a scanner) to align the image correctly when the paper has not been aligned correctly.</p> <p>Note This property only has any effect with devices that support such a feature.</p> <p>Default: true</p>															
SetTwainProgress	<p>ICV Twain setting for scanning.</p> <p>If this property is true, the device will display a progress bar during the direct (scan) acquisition of the image.</p> <p>Default: true</p>															
SetTwainThreshold	<p>ICV Twain setting for scanner.</p> <p>This property determines the dividing line between black pixels and white pixels for black and white scanning. It can take any value from 0 to 255, and for most devices, the default value is 128. A smaller value gives a whiter image and a larger value gives a blacker image.</p> <p>Default: 128</p>															
SICE.Enabled	<p>If set to true, the installation works in SignInfo Classic Edition (SICE) application mode.</p> <p>Default: false</p> <p>This parameter works in conjunction with the IsTeller parameter to set application/operation mode.</p> <p>The following table lists the possible combinations:</p> <table><tr><th>Installation</th><th>IsTeller</th><th>SICE.Enabled</th></tr><tr><td>Standard Thin Client</td><td>false</td><td>false</td></tr><tr><td>SICE Thin Client</td><td>false</td><td>true</td></tr><tr><td>Standard Teller</td><td>true</td><td>false</td></tr><tr><td>SICE Teller</td><td>true</td><td>true</td></tr></table>	Installation	IsTeller	SICE.Enabled	Standard Thin Client	false	false	SICE Thin Client	false	true	Standard Teller	true	false	SICE Teller	true	true
Installation	IsTeller	SICE.Enabled														
Standard Thin Client	false	false														
SICE Thin Client	false	true														
Standard Teller	true	false														
SICE Teller	true	true														
SiceResultList.MaxHeight	SignInfo image in SICE list															

Parameter	Description
	Default: 200
<code>SiceResultList.MaxWidth</code>	SignInfo image in SICE list Default: 400
<code>SiceResultList.ScalingStrategy</code>	SignInfo image in SICE list Default: 0
<code>SignatoryDetailsView.MaxHeight</code>	Signature in signatory details view Default: 100
<code>SignatoryDetailsView.MaxWidth</code>	Signature in signatory details view Default: 350
<code>SignatoryDetailsView.ScalingStrategy</code>	Signature in signatory details view Default: 0
<code>SignatoryListView.MaxHeight</code>	Signature in signatory list view Default: 200
<code>SignatoryListView.MaxWidth</code>	Signature in signatory list view Default: 200
<code>SignatoryListView.ScalingStrategy</code>	Signature in signatory list view Default: 0
<code>SignatoryNameClipLength</code>	Sets the maximum length of the signatory names in all drop down boxes in which they appear before they are clipped. The clipped portion of the name will then be replaced by "...". The length is specified in number of characters. Default: 30
<code>SignatoryNameClipLength.Teller</code>	Same as SignatoryNameClipLength , but only valid for all Teller modes. Overrides <code>SignatoryNameClipLength</code> .
<code>SignatoryNameClipLength.Teller.Kld2Teller</code>	Same as SignatoryNameClipLength , but only valid for Kld2Teller servlet interface. Overrides <code>SignatoryNameClipLength</code> and <code>SignatoryNameClipLength.Teller</code> .
<code>SignatoryNameClipLength.Teller.Ups2Teller</code>	Same as SignatoryNameClipLength , but only valid for Ups2Teller servlet interface. Overrides <code>SignatoryNameClipLength</code> and <code>SignatoryNameClipLength.Teller</code> .
<code>SignatoryNameSignatoryListClipLength</code>	Sets the maximum length of the rules text in the signatory list in which they appear before they are clipped. The clipped portion of

Parameter	Description
	<p>the text will then be replaced by "...". The Length is specified in number of characters.</p> <p>Default value: default value (set by cutoff attribute for SignatoryTag property singleRule in sigowentry-rules.jsp) is 0</p> <p>Default: 0</p>
SignatoryNameSignatoryListClipLength.Teller	<p>Same as SignatoryNameSignatoryListClipLength, but only valid for teller modes.</p> <p>Overrides SignatoryNameSignatoryListClipLength.</p>
SignatoryNameSignatoryListClipLength.Teller.Kld2Teller	<p>Same as SignatoryNameSignatoryListClipLength, but only valid for Kld2Teller servlet interface.</p> <p>Overrides SignatoryNameSignatoryListClipLength.Teller.</p>
SignatoryNameSignatoryListClipLength.Teller.Ups2Teller	<p>Same as SignatoryNameSignatoryListClipLength, but only valid for Ups2Teller servlet interface.</p> <p>Overrides SignatoryNameSignatoryListClipLength.Teller.</p>
SignCheckDisableImageTools	<p>Disables the image tools in SignCheck Clients i.e. Image transformation tools such as rotate, clean, invert in SignCheck Clients.</p> <p>Default value: false</p>
SignCheckDocumentBack.MaxHeight	<p>SignCheck Document Back</p> <p>Default: 300</p>
SignCheckDocumentBack.MaxWidth	<p>SignCheck Document Back</p> <p>Default: 400</p>
SignCheckDocumentBack.ScalingStrategy	<p>SignCheck Document Back</p> <p>Default: 0</p>
SignCheckDocumentFront.MaxHeight	<p>SignCheck Document Front</p> <p>Default: 300</p>
SignCheckDocumentFront.MaxWidth	<p>SignCheck Document Front</p> <p>Default: 400</p>
SignCheckDocumentFront.ScalingStrategy	<p>SignCheck Document Front</p> <p>Default: 0</p>
SignCheckDocumentSignature.MaxHeight	<p>SignCheck Document Signature</p> <p>Default: 200</p>
SignCheckDocumentSignature.MaxWidth	<p>SignCheck Document Signature</p> <p>Default: 400</p>

Parameter	Description
<code>SignCheckDocumentSignature.ScalingStrategy</code>	SignCheck Document Signature Default: 0
<code>SignCheckNotify</code>	Specify whether or not notification should be activated when a SignCheck queue becomes empty. Set to true, to activate notification. Default: false
<code>SignCheckNotifyApplet</code>	Specify if notification (when activated) is loaded via an applet or a self-reloading html page (if applets are not an option). See also SignCheckNotify property. Applet is always preferable. Default: true
<code>SignCheckNotifyInterval</code>	If the notification is activated, and is using a self-reloading html page instead of an applet, this parameter specifies the time span for reloading the page (and checking for a notification status update). The time span is configured in seconds. See also SignCheckNotify , SignCheckNotifyApplet properties. Default: 30
<code>SignCheckShowCheckData</code>	Specify whether or not the SignCheck document data should be displayed by default when a document is loaded. Default: false
<code>SignCheckShowHistoryData</code>	Set to true if item result history should be displayed in SignCheck view by default. Default value: true
<code>SignInfoCleanImagePreview</code>	In the SignInfo image view, hovering over the tree entries displays a preview of already loaded images. This property specifies whether or not the preview image should be "smoothed" before being sent to the browser. Setting this value to false will result in faster loading of the preview images but lower quality. Setting the value to true will result in better quality preview images, but slower performance. Default: false
<code>SignInfoImagePreview.MaxHeight</code>	SignInfo preview in tree Default: 150
<code>SignInfoImagePreview.MaxWidth</code>	SignInfo preview in tree Default: 150
<code>SignInfoImagePreview.ScalingStrategy</code>	SignInfo preview in tree Default: 0

Parameter	Description
<code>SignInfoShowDetailsOnRight</code>	<p>This parameter applies to the images displayed in the SICE application mode, and to the account images displayed as reference data in SignCheck.</p> <p>The property specifies whether image details should be shown on the right of the image instead of below the image. This is useful in situations where the screen resolution is small.</p> <p>Setting the value to true will result in image details being displayed to the right of the image.</p> <p>Setting the value to false will result in the image details being displayed below the image.</p> <p>Default: true</p>
<code>SignWithShortList.MaxHeight</code>	<p>Signature in infotip for 'Sign With' Short List</p> <p>Default: 80</p>
<code>SignWithShortList.MaxWidth</code>	<p>Signature in infotip for 'Sign With' Short List</p> <p>Default: 80</p>
<code>SignWithShortList.ScalingStrategy</code>	<p>Signature in infotip for 'Sign With' Short List</p> <p>Default: 0</p>
<code>SPTellerEntryStyle</code>	<p>CSS style sheet to be applied to all pages in all Teller servlet modes.</p> <p>Default value: <code>/jsp/styles/SignPlus-styles-s.css</code> or if commented out the setting of Default Style</p> <p>Note SPTellerEntryStyle overrides other Teller styles.</p>
<code>StoreVariantView.MaxHeight</code>	<p>Signature in Store Variant View</p> <p>Default: 300</p>
<code>StoreVariantView.MaxWidth</code>	<p>Signature in Store Variant View</p> <p>Default: 800</p>
<code>Teller.Request.Image.Transfer.Format</code>	<p>Describes the check image encoding for ICV Teller requests. Possible values are hexit (nibble hex) and base64.</p> <p>Default: base64</p>
<code>Teller.Search.RemoveZeros.AccountNo</code>	<p>Specifies whether leading zeros in the account number should be removed in the teller request.</p> <p>Default: false</p> <p>Important Both values for account number and for customer number should set to the same value in Account Model!</p>
<code>Teller.Search.RemoveZeros.CustomerNo</code>	<p>Specifies whether leading zeros in the customer number should be removed in the teller request.</p> <p>Default: false</p>

Parameter	Description
	Important Both values for account number and for customer number should set to the same value in Account Model!
TwainBottom	ICV Twain setting for scanner. The position of the bottom of the image, measured in TwainUnits from the top of the device. Value as Double (with point). Default: 7.3
TwainLeft	ICV Twain setting for scanner. The position of the left side of the image, measured in TwainUnits from the left side of the device. Example If TwainUnits is set to Inches and <code>TwainLeft = 1.5</code> an image acquired from a scanner will begin 1.5 inches from the left side of the scanner. Value as Double (with point). Default: 0
TwainResolution	ICV Twain setting for scanner. The resolution of the image in pixels (dots) per inch. If TwainResolution is set to an unsupported value, the nearest feasible value will be used instead. Default: 100.0
TwainRight	ICV Twain setting for scanner. The position of the right side of the image, measured in TwainUnits from the left side of the device. Value as Double (with point). Default: 17.5
TwainTop	ICV Twain setting for scanner. The position of the top of the image, measured in TwainUnits from the top of the device. Value as Double (with point). Default: 0
TwainUnits	ICV Twain setting for scanner. The units of measure to be used for any measurement related to the image being acquired. The property can take one of the following values: 0 (for Inches) 1 (for Centimeters) 2 (for Picas) 3 (for Points) 4 (for Twips)

Parameter	Description
	5 (for Pixels) Default: 0
Ups2TellerStyle	CSS style sheet to be applied to all pages in Ups2Teller servlet mode. Default: /jsp/styles/SignPlus-styles-s.css or if commented out the setting of Default Style
UseActiveX	Has to be set to 1 for usage of ActiveX controls with MS Internet Explorer. Default: 0
UseICV	Set to true if ICV (Day One Verification, see Release 4.0 Feature Description) should be enabled for the user. Default: false
UseLDAP	Set to 1 to enable LDAP login. Default: 0
useLeadTools	The usage of third party image library LeadTools can be enabled. Default: 0 (not enabled)
UserConfirmationForStoreAsGlobalFraud	Requires a confirmation when storing a global fraud signature in the blacklist. Since the user cannot delete a global fraud signature it is recommended that a user confirmation be required (=true). This setting is relevant only when 'EnableStoreAsGlobalFraud' is set. Default: true
UseSTV	Set to 1 if SignTeller Verification STV should be enabled. Default: 0
UseVpsv	Set to true if VPSV should be used as the default pad verification process for SignTeller Verification STV. Default: false
VBDQueueName	Specify the name of the VBD (Visual Branch Display) queue. Default: VBD
VpsvView.MaxHeight	Captured signature in VPSV view Default: 100
VpsvView.MaxWidth	Captured signature in VPSV view Default: 200
VpsvView.ScalingStrategy	Captured signature in VPSV view Default: 0

Parameter	Description
WatermarkFont	Allows the font to be used for the Watermark to be specified. The fonts installed on your system (server) where the Thin Client is installed are allowed. If the font is not located then Arial will be used as a default. Default: Arial
WatermarkFontBold	Insert the watermark using 'Bold' if specified. Default: false
WatermarkFontItalic	Insert the watermark using 'Italic' if specified. Default: false
WatermarkFontSize	Allows the font size to be used for the Watermark to be specified. Default: 20
WatermarkOpacity	Insert the watermark with the desired opacity level. Values from 20 to 100 are allowed. Increasing the setting increases the opacity of the watermark. Default: 50
WatermarkText	Configurable watermark text to be displayed. Default: Kofax FraudOne

Thin Client servlet parameters

The only servlet setting which can be changed is the setting for the InstanceName of the servlet when started. This provides the capability to create property files for this web server only. The Name is always 'InstanceName' and the value should be set to match the Instance Name set in the Administration Client.

The setting should follow the <web-app>... setting. An entry may already be in your web.xml file (commented out).

```
<web-app .....>
...
  <context-param>
    <param-name>InstanceName</param-name>
    <param-value>MyServerInstanceName</param-value>
  </context-param>
...
</web-app>
```

Engines parameters

The following table contains the description of the available Automat parameters.

Parameter	Description
AutoRuleVerification	Enable rules checking for given signatories without image comparison. Default: 0
BaseResolutionOrder	The resolution MinPelsOrder and MinPelsCleanedOrder are related to. Default: 200
BaseResolutionSignature	The resolution MinPelsSignature and MinPelsCleanedSignature are related to. Default: 200
BnoCurrency	BNO-specific currency
CheckCodes	List of all Form-Text-Codes for checks, e.g. 01, 02, 11
ConnectionRetries	Number of tests to the program stops. (-1 means unlimited retries) Default: 3
ConnectionRetryAlarm	Number of messages till an alarm is triggered and write a trace message. Default: 3
ConnectionRetryInterval	Time (in seconds) between occurs Critical messages Default: 30
DontCleanReference	Yes - Validation with not cleaned reference signatures Default: No (starting with SignPlus E II Release 3.6)
FeatureIDs	Possible FeatureID separated with "," 700,730,740,1025,1538,1539,1540,1543,1544,1545,1548,1793,1794,1795,1796,1797,1798,1799,1800
FormTypeList	Form types, for which specific amount limits and match rates should be used.
FSXResizeImg	SIVAL downscale image to the lowest resolution (of checks or reference) 1 - SIVAL resize 0 - without resizing Default: 0
GiroCodes	List of all Form-Text-Codes for giros, e.g. 17, 18, 19, 20, 21, 22, 23, 24, 50 ...
Host	Private Server Host Name or IP Address

Parameter	Description
	Default: localhost
Host4Notification	Hostname or IP address with notification server Default: localhost
HostSignBase	SignBase Server Host Name or IP Address Default: localhost
INICurrency	Initial currency setting for monetary amounts Default: EUR
Interval	Inactivity interval (in seconds) (waiting time before checking for entry) Update interval of monitor display Default: 30
IntlCodes	List of all Form-Text-Codes for international payment forms, e.g. 15
IRDMatchBonus	Match bonus for IRD's Default: 0
LicenseKey	License key for engines
Limit-1	Limit-1 = 200000: C1 For up to 200.000 Cents (2.000 EUR) a minimum match rate of at least C1 is required. The currency is according to the parameter INICurrency.
Limit-2	Limit-2 = 1000000: B5 Analog Limit-1
Limit22-1	Limit22-1 = 200000: A5 For up to 200.000 Cent (2.000 EUR) a minimum match rate of at least A5 is required, if the form is of type 22. The currency is according to the parameter INICurrency.
Limit22-2	Limit22-2 = 1000000: A3 Analog Limit22-1
Limit24-1	Limit24-1 = 200000: A5 For up to 200.000 Cent (2.000 EUR) a minimum match rate of at least A5 is required, if the form is of type 24. The currency is according to the parameter INICurrency.
Limit24-2	Limit24-2 = 1000000: A3 Analog Limit24-1

Parameter	Description
Limit-3	Limit-3 = 10000000: A5 Analog Limit-1
Limit-4	Limit-4 = 100000000: AA Analog Limit-1
LimitGIA-1	LimitGIA-1 = 90000000: C5 GIA Limits for BNO 001
LimitGIA-2	LimitGIA-2 = 950000000: B5 These Limits are not needed if AFD is not installed.
LimitGIA-3	LimitGIA-3 = 5000000000: A5 Analog LimitGIA-1
LimitGIA-4	LimitGIA-4 = 100000000000: AA For up to 100.000.000.000 Cent a minimum match rate of at least AA is required.
LoadRef2Engine	Which data needs the GIA client to process the checks 0 - no references load 1 - ref stockimages load 2 - ref templates load Default: 2
MaskAcctNo	Which part of the Account number is used. Used digits are marked with a +. Default: 00000000000000000000000000000000
MaskSerNo	Which part of the serial number is used. Used digits are marked with a +. Default: 00000000000000
MaxAmount	Monetary threshold amount for automatic verification: '1' = do not verify MaxAmount; Check up to the highest amount specified in Limit-x (see Limit-1) '0' = deactivate ASV; '>0' = maximum amount allowed for ASV Should only be defined in section [Common] ! Default: -1 With the following (maximum of 10) entries, fine-tuning (calibration) is set in the introduction phase for decisions related to internal bank risk threshold values, taking into consideration technical (e.g. image resolution) and semantic quality (e.g. age of signatures).

Parameter	Description
	AAA = no ASV / APSV AA = highest rating F4 = lowest rating Note ASV and AFD must always be identical. ASV and APSV can have individual settings.
MaxSimplicity	Upper limit for the simplicity in the structure of a reference resp. check signature. Maximum: 90 Default: 70
MinPelsCleanedOrder	Similar for the cleaned signature Default: 400
MinPelsCleanedSignature	Similar for the cleaned signature Default: 400
MinPelsOrder	Minimum number of image points in the document signature (not cleaned) for automatic verification (min. 400!) Default: 400
MinPelsSignature	Minimum number of image points in the reference signature (not cleaned) for automatic verification (min. 400!) Default: 400
MinResolution	Minimum resolution of reference signatures and snippets for automatic verification Default: 150
NotLatin	Specifies whether only signatures in Latin script are to be verified. Yes - signatures that are not in Latin alphabet will be verified Default: No
NotVerified	Specifies whether the checks of unverified customers are to be processed in the machine. No - only accounts of verified customers are checked by the automats. Default: No
NumFsxSplitsBeside	Maximum number of vertical SIVAL splits. If this parameter is not specified: - if SepStepBeside is 0, this parameter is set to 0 too - if SepStepBeside is greater 0, this parameter is set to 2
NumFsxSplitsUpon	Maximum number of horizontal SIVAL splits.

Parameter	Description
	<p>If parameter not specified:</p> <ul style="list-style-type: none"> - if SepStepUpon is 0, this parameter is set to 0 too - if SepStepUpon is greater 0, this parameter is set to 2
ParamCount	<p>Count of GIA parameter</p> <p>Default: 0</p>
ParameterFile	RecoStar ConfigFile name
ParamName-1	<p>Name of parameter 1</p> <p>For FeatureID 700: Name of parameter for default cropping area</p> <p>For FeatureID 730: Name of parameter for clean level</p> <p>For FeatureID 740: Name of parameter for default searching area</p> <p>For FeatureID 1025: Name of parameter is CSDetail</p> <p>For FeatureID 1541: Name of parameter is DateType</p> <p>For FeatureID 1548: Name of parameter is PayeeRegion</p> <p>For FeatureID 1793, 1794, 1802: Name of parameter is SerialNumberRange</p> <p>For FeatureID 1796, 1797, 1805: Name of parameter is MinStatisticalCycles</p> <p>For FeatureID 1804: Name of parameter is AmountRange</p>
ParamName-2	<p>Name of parameter 2</p> <p>For FeatureID 700: Name of parameter for FormType-dependent cropping area</p> <p>For FeatureID 740: Name of parameter for FormType-dependent searching area</p> <p>For FeatureID 1541: Name of parameter is YearRange</p> <p>For FeatureID 1548: Name of parameter for FormType-dependent searching area</p> <p>For FeatureID 1796: Name of parameter DiffMinValues</p> <p>For FeatureID 1797, 1805: Name of parameter DiffMaxValues</p> <p>For FeatureID 1804: Name of parameter is MinStatisticalCycles</p>
ParamName-3	<p>Name of parameter 3</p> <p>For FeatureID 1541: Name of parameter is DateHeuristics</p> <p>For FeatureID 1797, 1805: Name of parameter is DiffAverageValue</p> <p>For FeatureID 1796: Name of parameter is DiffMaxValue</p>
ParamName-4	<p>Name of parameter 4</p> <p>For FeatureID 1541: Name of parameter is TwoDigitYear</p>

Parameter	Description
	For FeatureID 1796: Name of parameter is DiffTotalValue For FeatureID 1797, 1805: Name of parameter is DiffCalAverageValue
ParamName-5	Name of parameter 5 For FeatureID 1796: Name of parameter is DiffAverageValue For FeatureID 1797: Name of parameter is DiffCalAverageValue
ParamName-x	Name of parameter x
ParamString-1	Value of parameter 1 For FeatureID 700: Value for the default cropping area For FeatureID 730: Value for the default cropping area (Range 1-999) For FeatureID 740: Value for the default searching area For FeatureID 1793: Value for range expansion, on which the Serial number is near For FeatureID 1794: Value for range expansion, on which the Serial number is near For FeatureID 1796, 1797, 1805: Number of minimum statistical cycles to be able to do a verification.
ParamString-2	Value of parameter 2 For FeatureID 700: Value for the cropping area For featureID 740: Value for the searching area For FeatureID 1796: Defines "how far" (percentage) an amount is from minimum amounts to be considered "critical". Range: 0-100 For FeatureID 1797: Defines percentage value how much the difference limit is above highest. Range: 0-100
ParamString-3	Value of parameter 3 For FeatureID 1796: Defines "how far" (percentage) an amount is from maximum amounts to be considered "critical". Range: 0-100 For FeatureID 1797: Defines percentage value how much the difference limit is above average. Range: 0-100
ParamString-4	Value of parameter 4 For FeatureID 1796: Defines "how far" (percentage) an amount is from average amounts to be considered "critical". Range: 0-100

Parameter	Description
	For FeatureID 1797: Defines the percentage for the maximum increase of the limit for average Range: 0-100
ParamString-5	Value of parameter 5 For FeatureID 1796: Defines the percentage for the maximum increase of the limit for average Range: 0-100
ParamString-x	Value of parameter x
Port4Notification	Port for notification Default: 0
PortSignBase	For AVC SignBase Server Port Default: 0
Queue4Notification	Queue no. in which the request data are saved Default: 0
Register4Notification	Client register for a notification to process day one requests.
RemoveInvalidVariants	Removal of invalid variants. Default: Yes
RemovePrefixedDate	Removal of date placed in the signature area Default: No
SepMinBorder	Minimum margin in % (0%..40%) Default: 20
SepStepBeside	Vertical shift steps in % to search for the second signature. Default: 10
SepStepUpon	Horizontal shift steps in % to search for the second signature. Default: 10
SplitSingleSignerAccounts	Signatures are only split if value is TRUE. For customer who has only single signer accounts, it is not recommendable split signatures (to many wrong decisions). Default: TRUE
StatValueWriteInterval	Interval to write statistic values Range: 100 - 20000 items Default: 100
TimeFontSize	Font size of time statement

Parameter	Description
TwinTestRating	Maximum similarity allowed between reference signatures. Allowable values are AA to F4. The TwinTest is deactivated through the entry AAA or F5. Default: 0
UseDefaultFraud	Default frauds will check in front of normal checking. Default: No
UseLimits4DynVer	No - The default value (threshold 80) of the engine is used. 0 - 79 not accepted 80 - 100 accepted Yes - The Limits in the APSV section are used to decide for accepted or not accepted.
UseOnlyCleanSnippets	Validate only cleaned signature snippets from documents (might make sense in countries with stamps in the signature areas) Default: No
UsePayeeList	The GIA automat sends a list with payee names for feature ID 515 to the engines. Default: No
UseRestInst	No - Instructions and restrictions won't be considered. Yes - If there is a restriction or instruction present for the account, the customer or the signatory the document will be rejected with returncode 14 (instruction on customer / account) or 15 (instruction on signatory) and queued to VSV. The Customer 'C', Account 'A' and / or Signatory 'S' evaluation of instructions and restrictions could also be switched on / off separately by specifying C, A and / or S separated by commas. Exceptions Customer / Account: Restriction 1 and 2 won't be rejected. Signatory: Instruction 4 (in case of a check), 5, 71 (in case of a giro), 6 (in case of a cross border/intl payment form) and 72 (in case of a check or a giro) won't be rejected. For the signatory instruction codes 4, 5, 6, 71, 72 also specify the payment form's Form-Text-Code. Default: No
VerifyDynamic	No - Only a static verification will be done. Default: Yes
VerifyEndorsementSignature	Yes - External checks are processed and identified as such. Default: No

Parameter	Description
Wait4Notification	Maximum reaction wait time (in seconds) Default: 10
WorkQueue	Automat clients search in this Queue for work Default: 0
WriteStat	Yes - The Automat client writes statistic information Default: Yes

Archive Interface Server parameters

The following table contains the description of the available Archive Interface Server parameters.

Parameter	Description
AccountFormat	Extract account/customer number with a fix length and a fix position. Example Filename: 123456xx <pre>AccountNoDigitPlaceholder = # WildcardPlaceholder = ? AccountFormat = #####??</pre> Extracted account number is 123456.
Accountno.minsize	Minimum number of digits to be entered
Accountno.size	Maximum number of digits to be entered
AccountNoDigitPlaceholder	Placeholder for a digit of the account number in AccountFormat.
AllowSbCreateCustomer	1 - If a new account can be created due to the presence of one or more images then set to 1. 0 - If set to 0 then an account must be present before saving of an image is possible (done in SignBase client). Default: 1
AmsServer.MaxThreads	Specifies the maximum number of image servers (or threads) which may be run simultaneously. Default: 10
AppendImagesToXML	This setting will have an effect only when image files and prepared XML data is available for the same customer/account.

Parameter	Description
	<p>0 - The XML data and the image files located will be sent in separate packages. This means that a minimum of 2 client requests are required if both XML data and image files are available.</p> <p>1 - If both XML data and image files are located for the same customer/account then the AIS will append any separate image files to the located XML data before sending it to a client. This may cause certain items located in the original XML data such as ActionCode, etc. to be changed.</p> <p>Default: 0</p>
Bankcode.minsize	Minimum number of digits to be entered
Bankcode.size	Maximum number of digits to be entered
Branchcode.minsize	Minimum number of digits to be entered
Branchcode.size	Maximum number of digits to be entered
ClassPath	
Custom.AccountModel	<p>Some cases require that although the Customer data model is used in SignPlus that the data provided as input for the AIS can only be provided in Account Model. This setting allows the AccountModel setting from the custom.properties configuration file located in the custom.zip to be overridden (for the AIS only!).</p> <p>Note This is used only in special cases and will require special handling within the client. Do NOT use this setting unless it has been specified by Kofax.</p> <p>Default: value located in custom.zip</p>
Customerno.minsize	<p>Minimum number of digits to be entered.</p> <p>For account model set to same size as Accountno.minsize.</p>
Customerno.size	<p>Maximum number of digits to be entered.</p> <p>For Account Model set to same size as Accountno.size.</p>
DeleteInvalidImageFiles	<p>Enables the server to remove invalid image files.</p> <p>Default: OFF</p>
ElapsedLockedForUserTime	<p>Once items have been locked for a specific user they will automatically be released and available for all users after this time has elapsed. This entry contains the time (in minutes) after which the locked items will be released.</p> <p>Default: 1440 (approx. 24 hours)</p> <p>Maximum: 34560 (approx. 24 Days)</p>
ElapsedOnHoldTime	<p>Once items have been placed 'On Hold' they will automatically be released and available for all requests after this time has elapsed. This entry contains the time (in minutes) after which the 'On Hold' items will be released.</p>

Parameter	Description
	<p>Default: 10080 (approx. 7 days)</p> <p>Maximum: 34560 (approx. 24 days)</p>
EmbeddedAccountFormat	<p>Extract embedded account/customer number with a variable length and position.</p> <p>Example</p> <p>filename: P123456C0</p> <p>EmbeddedAccountNoPlaceholder = #</p> <p>EmbeddedAccountFormat = P#C</p> <p>Extracted account number is 123456.</p>
EmbeddedAccountNoPlaceholder	Placeholder for account number in EmbeddedAccountFormat
FormattedFileName	<p>Supported by Version 3.1.13 and higher only.</p> <p>Using a formatted file name allows certain default values to be overwritten/specified within the image file name.</p> <p>Presently wild cards (x) and the BNO (n) may be specified within the formatted file name.</p> <p>If a formatted file name is specified then it is valid for all files read and processed by the ImageLocators.</p> <p>Valid for input XML files also but the BNO contained within the XML will NOT be verified to match the BNO specified within the formatted file name.</p> <p>If not used then the BNO will be assigned as specified within the client custom.properties as always.</p> <p>Example</p> <p>'x.n.x.x' means:</p> <p>The values located from the beginning of the name to the first separator (.) is reserved for the customer/account number.</p> <p>The values between the 1st and 2nd '.' Will be used as the BNO.</p> <p>Any file names that do not match the specified format will not be processed.</p> <p>Default: empty (unused)</p>
GroupByFileName	<p>This setting can be activated when UseFileNameAsAccount=0 to specify that all images which have the same file name (but different extensions) belong to the same account.</p> <p>0 - No, images will be sent individually</p> <p>1 - Yes, images will be sent grouped by the file name</p>
ImageLocatorEnabled	<p>Activates the Image Locator. If activated the specified folders will be monitored for new image file types as specified.</p> <p>0 - Off</p> <p>1 - On</p>

Parameter	Description
	Default: 1
ImageLocatorPath	Specifies the folders to be monitored by the Image Locator. All folder must be input fully qualified and end with ';'. If multiple folders are desired then the next folder begins after the delimiter (;).
ImageLocatorPause	A pause time can be specified for the Image Locator of the AIS. After all folders / files have been checked the Image Locator will wait the specified time (in minutes) before restarting. Default: 10
ImageLocatorTypes	Specifies the file types to be monitored within the specified folders (ImageLocatorPath). All file types must end with ';'. If multiple file types are desired then the next file type begins after the delimiter (;).
ImageServer.MaxThreads	The number of processing threads allowed to process client requests can be specified. This setting should only be confirmed as specified by Kofax. Please check with support before modifying this value. Allowed: 1 - 25 Default: 5
JDBC.DefaultConnection.Catalog	Name of the database to which is connected (valid for MS SQL Server only)
JDBC.DefaultConnection.DbPassword	Database user password (i.e. signplus). Empty for default user password.
JDBC.DefaultConnection.DbURL	Name and location of the database i.e. for DB2: jdbc:db2:signplus i.e. for Oracle: jdbc:oracle:thin:@system_name:1521:signplus (1521 -> port number)
JDBC.DefaultConnection.DbUser	Database user (i.e. signplus). Empty for default user.
JDBC.DefaultConnection.Driver	Location of the driver class i.e. for DB2: COM.ibm.db2.jdbc.app.DB2Driver i.e. for Oracle: oracle.jdbc.driver.OracleDriver
JDBC.LoadDefaultDriver	Use the JDBC connection 0 - No 1 - Yes (required)

Parameter	Description
	Default: 1
MainClass	
MaxImagePackageSize	<p>Restricts the maximum number of images sent to a client. This is used only to restrict the actual number of images received per request.</p> <p>If more images are available they will be sent in the next request until all have been processed. Multiple clients may receive data pertaining a single customer or account.</p> <p>Default: 4</p>
Name.Size	<p>For server maintenance console only! Specifies the size of the file name incl. path.</p> <p>Default: 255</p>
Program	
ReplaceAllImagesOnUpdate	<p>When image data is transferred to the SignPlus client it is possible to specify that the images presently being sent to the requesting client are to replace all current images which are presently saved for this customer / account.</p> <p>If this functionality is desired then set to 1. Otherwise all data sent will be appended to the current information for the customer / account.</p> <p>Default: 0</p>
Server.AcceptClientRequests	<p>Allows the current Archive Interface Server to accept and process client requests. This is always required, unless the only console is to be used for maintenance purposes.</p> <p>When starting with this set to 0 (No) all locators should be turned off prior to starting. This prevents the registration of unwanted files in the status table.</p> <p>0 - No 1 - Yes Default: 1</p>
Server.AllowDbAdmin	<p>When the Archive Interface Server console is activated menu items to provide some database administration features such as create DB. If turned on then Server.AllowDbManagement will automatically be turned on.</p> <p>0 - No 1 - Yes</p>
Server.AllowDbManagement	<p>When the Archive Interface Server console is activated menu items to provide some database management (delete, reset,...) will appear (if YES). This is automatically turned on if Server.AllowDbAdmin is on.</p> <p>0 - No 1 - Yes</p>

Parameter	Description
<code>Server.ConsoleEnabled</code>	<p>Allows the server maintenance console to be displayed when starting. The console is only for administrators.</p> <p>0 - No 1 - Yes Default: 1</p>
<code>Server.Port</code>	<p>Specifies the port for all socket connections to the AMS server.</p> <p>Default: 2015</p>
<code>Server.ShowTrace</code>	<p>When the Archive Interface Server is started the trace window may automatically be opened and remain on until shut off manually</p> <p>0 - No 1 - Yes</p>
<code>Server.TraceLevel</code>	<p>The trace level is a combination (bitwise OR) of the following values.</p> <p>-1 - trace everything 0 - no trace at all 1 - error messages 2 - warnings 4 - debugging information 8 - information 16 - resource information 32 - SQL Trace</p> <p>Note Minimum TraceLevel=11 (errors, warnings, information)</p>
<code>Server.UseHostDB</code>	<p>Set to 1 when the database is located on a host. Different (smaller) column sizes will be used for various entries. For REL-3-1-8 and higher. (Valid presently on for DB2)</p> <p>0 - No 1 - Yes Default: 0</p>
<code>SortGetNextBy</code>	<p>When requests are received from the client the AIS can sort the list of items to be processed by:</p> <p>IMG_DATE - Sorting will be performed based of the date of the actual data file.</p> <p>IMG_LOCATION - Sorting will be performed based of the name (including the path) of the actual data file.</p> <p>TIME_STAMP - Sorting will be performed based of the time stamp of the record. The timestamp reflects when the item was registered by the AIS.</p> <p>Note Input will be checked case sensitive!</p>

Parameter	Description
	Default: TIME_STAMP
SortGetNextOrder	<p>Sorting order desired for list of items to be processed upon the GetNext client request:</p> <p>ASC - The oldest entry from the result list will be sent as a reply to the client request.</p> <p>DESC - The newest entry from the result list will be sent as a reply to the client request.</p> <p>Note Input will be checked case sensitive!</p> <p>Default: ASC</p>
StatisticsInterval	<p>Interval (in seconds) after which license statistics are sent to the server.</p> <p>Default: 900</p>
StatisticsInterval.FirstTime	<p>Interval (in seconds) after which the first license statistics are sent to the server.</p> <p>Default: 120</p>
Timestamp.Size	<p>For server maintenance console only!</p> <p>Specifies the size of the timestamp.</p> <p>Default: 24</p>
UseFileNameAsAccount	<p>When scanning for images the customer/account number is determined from the actual file name located. This is the standard method. If this is not the case this should be set to 0 and each request from the client will require the user to input the actual customer / account number for the image. If set to 0 then all images from that point on will be marked as 'unknown' customer account.</p> <p>This setting has no effect on located XML files, the customer / account number is clearly specified in this case.</p> <p>Default: 1</p>
UseLockingForUser	<p>This feature locks all items from the same customer/account sent to a user for this user. Once items have been locked for a specific user they will not be sent to other users. The current user will receive the next set of images automatically when the next data set is requested. If not requested by the current user they will be released after the setting 'ElapsedOnHoldTime' has elapsed.</p> <p>Default: 0</p>
UsePlaceOnHold	<p>Items may be placed on hold if desired. The client requests that this item(s) and all further data for this customer/account be placed on hold. A special request/confirmation is required from the client.</p> <p>Note This is used only in special cases and will require special handling within the client. Do NOT use this setting unless it has been specified by Kofax.</p>

Parameter	Description
	Default: -
UseProtocol	<p>Activates support for creating protocol records for monitoring the AIS processing. This feature requires AIS version 3.2.2 or higher. Other client features may be required to obtain data registered in the protocol table.</p> <p>Contact Kofax before activating.</p> <p>0 - off</p> <p>Default: 0</p>
UseSPQualifier	<p>For server maintenance console only! Specifies the database qualifier default to the SignPlus Data Exchange Base (DEB).</p> <p>Default: DEB (for SignPlus DB2)</p>
WildcardPlaceholder	Placeholder for any character being not part of the account number.
XMLComplimentsImages	<p>This setting allows ASCII data to be provided in an XML file and if any image files are located for the same customer/account then the images will be placed into any AccountImages found in the XML data where no image data is located. If no AccountImage located without image data then a new AccountImage for the current customer/account will be created (as normal).</p> <p>0 - Always create a new AccountImage structure in the customer/account for the image files located.</p> <p>1 - Check if AccountImage data is present without image data. If so, insert any image data located into these AccountImage fields, else create new AccountImage data.</p> <p>Note This is used only in special cases. Do NOT use this setting unless it has been specified by Kofax.</p> <p>Default: 0</p>
XmlLocatorEnabled	<p>Activate the XML Locator. If activated the specified folders will be monitored for new XML files as specified.</p> <p>0 - Off</p> <p>1 - On</p> <p>Default: 1</p>
XmlLocatorPath	Specifies the folder (ONLY 1) to be monitored for Kofax XML files.
XmlLocatorPause	<p>A pause time can be specified for the XML Locator. After all XML files have been checked the locator will wait the specified time (in minutes) before restarting.</p> <p>Default: 10</p>
XmlLocatorType	Specifies the file extension to be monitored within the specified folder (XmlLocatorPath). Normally only XML is desired, but an

Parameter	Description
	alternative extension is allowed here. The file content must be a Kofax XML file. Default: XML

Port number defaults

Portnumber	Description
2000	SignBase
2001	SignCheck
2002	ASV
2003	Logger (no longer used; E I only)
2004	Monitor request
2005	Monitor log
2006	AFD (till Release 3.7)
2007	SDQ
2008	APSV
2010	Teller interface / NOS
2011	Logon
2012	SignBase Physical Verification for Inwards Clearing
2013	Twain
2014	Workflow Notification
2015	Archive Interface Server (Communication)
2016	Archive Interface Server (Statistics)
2017	Workflow server
2018	Messages to WF router (UDP)
2019	Connection between CRS engine and WF router (TCP)
2020	Network License Manager

Portnumber	Description
2021	GIA
2022	Schedule Server

FraudOne Server messages

In the following chapters you will find a list of the server messages for [SignBase](#), [SignCheck](#), [ASV](#), [AFD](#), and [STV](#).

Messages for SignBase Servers

- 00 Logon
- 01 Result : logon
- 02 Logoff
- 03 Result : logoff
- 10 Get whole customer
- 11 Get some signatures
- 12 Get all users
- 14 Get all user actions (Protocol)
- 15 Search for customer
- 16 Search for account
- 17 Search for customers not verified
- 18 Search for customers with unassigned variants
- 19 Search for non processed account images
- 1A Insert/update/delete object (except signature and user)
- 1B Insert/update/delete signature
- 1C Verify
- 1D Insert/update/delete user/powers
- 1E Copy signatories
- 1F Get documents / masks
- 20 Result : all objects except signature
- 21 Result : signature
- 22 Result : Get all user actions

- 25 Result : found customer
- 26 Result : found account
- 27 Limit exceeded
- 2A Result : object (inserted/updated/deleted)
- A8 Report Server messages
- A9 Non-customer data handling messages
- AA All configuration management messages

Messages for SignCheck Servers

- 50 Get next
- 51 Result: next
- 52 Get specific
- 53 Result: cheque data
- 54 Write result
- 55 Result written
- 56 Get SC list
- 57 Get workflow list
- 58 Get SC status
- 59 Result: SC status
- 5A Get status history for document
- 5B Result: document status history
- 5C Get workflow queue names
- 5D Result: workflow queue names
- 5E Change order data of form
- 5F Result: SC list
- 60 Get decisions of users
- 61 Result: decisions of users
- 62 Update general SignCheck statistics

Messages for ASV and AFD Servers

- 80 Get .Ini file
- 81 Result: .Ini file
- 82 Statistics SCA
- 8A Get next for SignCheck automat
- 8C Update SignCheck automat
This message type is, if applicable, specified for only one specific ASV or AFD server in order to activate run time optimization.

Messages for STV Servers

- 30 Client request to STV-Server
- 31 Client result to STV-Server
- 32 Get next for APSV automat
- 33 APSV-Signature data
- 34 WriteResultAPSV
(sends the result of a static and/or dynamic compare from APSV-Client to APSV-Server)
- 35 WriteStatisticsAPSV
(sends the statistics from APSV-Client to APSV-Server)
- 36 APSV-Returncode
(if the APSV-Server has already been able to process the request)

GIA configuration for PreProcessingEngine

Purpose

In order to support new engines and additional engine features in an easily configurable way a new automat type will be introduced starting with Release 3.9. The new automat is called GIA which stands for **G**eneral **I**mage **A**alyzer. The goal is to have a new automat architecture that allows

- to plug in new engines easily via a common engine API
- to provide a possibility to configure different GIA engines for different purposes
- to have only one GIA server that different engines can connect to regardless of configured engine features

Every engine contains a set of features, that must be configured as required in the configuration **automat2.ini**.

Parameters are described in the Parameter Overview of the Appendix under [Parameter Engines](#).

The configuration can be accomplished in the following configuration area.

A section **[FeatureParms-x]**, here the parameters for every single FeatureID are defined, because not everybody needs special parameter settings.

Example

FeatureParm-700
in dependence of count of FeatureIDs in [GIA] section

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount =
ParamName-1 =
ParamString-1 =
ParamName-2 =
ParamString-2 =
ParamName-x =
ParamString-x =
```

Possible feature overview

The following FeatureIDs are supported by the PPE Engine:

FeatureID	Description
700	Cropping signatures Cut out a signature from a check. The cropping area must be defined inside the range of the check and we can have different cropping areas based on check types.
730	Cleaning signatures Cleaning signatures tries to reduce any pixels that do not belong to the signature. Therefore it increases the quality of the signatures and delivers better results on signature comparison.
740	Searching signatures Try to identify a signature in the searching area of the check and if found one, cut it out. The searching area must be defined inside the range of the check and we can have different searching areas based on check types.
750	Splitting signature - not yet implemented

Configuration of the PPE features

Parameters for all features

For cropping, cleaning and searching a check image is required in the engine.

[GIA]

This section specifies the parameters for GIA.

See chapter [The configuration file automat2.ini](#) for information on general parameters.

```
FeatureIDs = 700,730  
LoadRef2Engine = 0
```

FeatureID 700 - Cropping signatures

For cropping signatures only a check is required in the engine.

[FeatureParm-700] - Parameter specification for Cropping

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 2  
ParamName-1 = CROP1Default  
ParamString-1 = 1,1,100,100  
ParamName-2 = CROP1-xxx  
ParamString-2 = 1,1,100,100
```

FeatureID 730 - Cleaning the cropped signature

For cleaning a cropped signature is required in the engine.

[FeatureParm-730] - Parameter specification for Cleaning

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 1  
ParamName-1 = CleanLevel  
ParamString-1 = 970
```


FeatureID 740 - Searching signatures

For Searching signatures we need only a check in the engine.

[FeatureParm-700] - Parameter specification Searching

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 2
ParamName-1 = CROP0Default
ParamString-1 = 1,1,100,100
ParamName-2 = CROP0-xxx
ParamString-2 = 1,1,100,100
```

Example of a GIA PPE configuration

This is an example of a PreProcessingEngine use case with GIA and ICV (Instant Check Verification).

ICV puts the check to queue 93 and the GIA automat receives a notification for work. Then the GIA gets the check and puts it to the PPE engine.

In the PPE engine the FeatureIDs 740 and 730 are processed. Therefore is a searching area defined.

If the search process can't find a signature, the default cropping area of the FeatureID 700 is used.

If the customer uses FormType, the process searches in the parameter for searching / cropping area with the FormType (in the example the FormType is "S10" and "S12").

```
[GIA]
Register4Notification = Yes
Port4Notification = 2014
Queue4Notification = 93
Wait4Notification = 2

FeatureIDs = 740,730

LoadRef2Engine = 0    ;-- 0 = no references load

[FeatureParam-700]
ParamCount = 3
ParamName-1 = CROP1Default
ParamString-1 = 290,412,488,508    ;left,top,right,bottom
ParamName-2 = CROP1-S10
ParamString-2 = 705,296,1091,406
ParamName-3 = CROP1-S12
ParamString-3 = 705,296,1091,406

[FeatureParam-730]
ParamCount = 1
ParamName-1 = CleanLevel
ParamString-1 = 977

[FeatureParam-740]
```

```
ParamCount = 2
ParamName-1 = CROP0Default
ParamString-1 = -632,-326,-32,-100
ParamName-2 = CROP0-S10
ParamString-2 = 705,296,1091,406
```

GIA configuration

Purpose

In order to support new engines and additional engine features in an easily configurable way a new automat type will be introduced starting with Release 3.9. The new automat is called GIA which stands for **General Image Analyzer**. The goal is to have a new automat architecture that allows

- to plug in new engines easily via a common engine API
- to provide a possibility to configure different GIA engines for different purposes
- to have only one GIA server that different engines can connect to regardless of configured engine features

Every engine contains a set of features, that must be configured as required in the configuration **automat2.ini**.

Parameters are described in the [Server Manager Parameter](#) chapter.

Two configuration areas can be distinguished:

1. A section **[EngineParms]**; general engine parameters are defined here.

This section specifies the parameters for GIA engine only.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount =
ParamName-1 =
ParamString-1 =
ParamName-2 =
ParamString-2 =
ParamName-x =
ParamString-x =
LicenseKey =
```

2. A section **[FeatureParms-x]**, here we define parameters for every single feature if applicable.

Example

FeatureParm-1538

dependent on number of features in [GIA] section

This section specifies the feature parameters for the engine only.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount =
ParamName-1 =
ParamString-1 =
ParamName-2 =
ParamString-2 =
ParamName-x =
```

[ParamString-x](#) =

GIA configuration for Statistical Analysis Engine (SAE)

Purpose

In order to support new engines and additional engine features in an easily configurable way a new automat type will be introduced starting with Release 3.9. The new automat is called GIA which stands for **General Image Analyzer**. The goal is to have a new automat architecture that allows

- to plug in new engines easily via a common engine API
- to provide a possibility to configure different GIA engines for different purposes
- to have only one GIA server that different engines can connect to regardless of configured engine features

Every engine contains a set of features, that must be configured as required in the configuration **automat2.ini**.

Parameters are described in the Parameter Overview of the Appendix under [Engine Parameters](#).

The configuration can be accomplished in the following configuration area.

A section **[FeatureParms-x]**, here the parameters for every single FeatureID are defined, because not everybody needs special parameter settings.

Example

FeatureParm-1793

in dependence of count of FeatureIDs in [GIA] section

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount =
ParamName-1 =
ParamString-1 =
ParamName-2 =
ParamString-2 =
ParamName-x =
ParamString-x =
```

Possible feature overview

The following FeatureIDs are supported by SAE Engine:

FeatureID	Description
1793	Check serial number verification in Issued Serial Number ranges Verify that the check serial number is within or near the issued number range. What is denoted as near defines the Feature parameter SerialNumberRange.
1794	Check serial number verification in Observed Serial Number ranges Verify that the check serial number is within or near the observed number range. What is denoted as near defines the Feature parameter SerialNumberRange.

FeatureID	Description
1795	<p>Check serial number verification duplicate check</p> <p>Checks if the check serial number has already been processed in the past, including the current day.</p>
1796	<p>Check amount verification</p> <p>Checks if the check amount is within all boundary conditions or whether the amount has already been processed on that day and whether average amount increased by critical amount.</p>
1797	<p>Check velocity verification</p> <p>Checks if the check velocity (numbers of checks processed per cycle/interval) is within all boundary conditions or whether highest number of checks per day exceeded by critical amount or whether average number of checks per day exceeded by critical amount or whether average number of checks per day increased by critical amount.</p>
1798	<p>Check amount highlighting</p> <p>The check amount highlighting feature allows patterns to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.</p>
1799	<p>Account number highlighting</p> <p>The account number highlighting feature allows account numbers or account number fragments to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.</p>
1800	<p>Branch / transit code highlighting</p> <p>The branch / transit code highlighting feature allows branch codes or branch code fragments to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.</p>
1802	<p>Check serial number verification in Observed Serial Number ranges for external accounts</p> <p>Verify that the check serial number is within or near the observed number range. What is denoted as near defines the Feature parameter SerialNumberRange.</p>
1803	<p>Check serial number verification duplicate check for external accounts</p> <p>Checks if the check serial number has already been processed in the past, including the current day</p>
1804	<p>Check amount verification for external accounts</p> <p>Checks if the check amount is within all boundary conditions or whether the amount has already been processed on that day and whether average amount increased by critical amount.</p>
1805	<p>Check velocity verification for external accounts</p>

FeatureID	Description
	Checks if the check velocity (numbers of checks processed per cycle/interval) is within all boundary conditions or whether highest number of checks per day exceeded by critical amount or whether average number of checks per day exceeded by critical amount or whether average number of checks per day increased by critical amount.
1806	Check amount highlighting for external accounts The check amount highlighting feature allows patterns to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.
1807	Account number highlighting for external accounts The account number highlighting feature allows account numbers or account number fragments to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.
1808	Branch / transit code highlighting for external accounts The branch / transit code highlighting feature allows branch codes or branch code fragments to be specified and to generate a result that highlights the check (marks it as a check with a higher risk) that an analyst might want to look at in more detail.

Configuration of the SAE features

Parameters for all features

To process the features it is necessary that check data and the statistical data are present.

[GIA]

This section specifies the parameters for GIA.

See chapter [The configuration file automat2.ini](#) for information on general parameters.

```
FeatureIDs = 1793,1794,1795,1796,1797,1798,1799,1800
LoadRef2Engine = 2
```

FeatureID 1793 - Serial number verification in issued serial number ranges

To process the features for "Check serial number verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1793] - Parameter specification for serial number verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 1
```

```
ParamName-1 = SerialNumberRange  
ParamString-1 = 100
```

FeatureID 1794 - Serial number verification in observed serial number Ranges

To process the features for "Check serial number verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1794] - Parameter specification for serial number verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 1  
ParamName-1 = SerialNumberRange  
ParamString-1 = 100
```

FeatureID 1795 - Serial number duplicate verification

To process the features for "Check serial number verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1795] - Parameter specification for serial number verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1796 - Amount range verification

To process the features for "Check amount verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1796] - Parameter specification for amount range verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 5  
ParamName-1 = MinStatisticCycles  
ParamString-1 = 30  
ParamName-2 = DiffMinValue  
ParamString-2 = 5 ; 5%  
ParamName-3 = DiffMaxValue  
ParamString-3 = 2 ; 2%  
ParamName-4 = DiffTotalValue  
ParamString-4 = 4 ; 4%  
ParamName-5 = DiffAverageValue  
ParamString-5 = 5 ; 5%
```

FeatureID 1797 - Check velocity verification

To process the features for "Check velocity verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1797] - Parameter specification for check velocity verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 4
ParamName-1 = MinStatisticalCycles
ParamString-1 = 30 ; e.g. 30 days
ParamName-2 = DiffMaxValue
ParamString-2 = 5 ; 5 %
ParamName-3 = DiffAverageValue
ParamString-3 = 4 ; 4 %
ParamName-4 = DiffCalAverageValue
ParamString-4 = 5 ; 5 %
```

FeatureID 1798 - Check amount highlighting

To process the features for "Check amount highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1798] - Parameter specification for amount highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1799 - Account number highlighting

To process the features for "Account number highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1799] - Parameter specification for account number highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1800 - Branch/transit code highlighting

To process the features for "Branch/Transit code highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1800] - Parameter specification for branch/transit code highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1802 - Serial number verification in observed serial number ranges for external accounts

To process the features for "Check serial number verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1802] - Parameter specification for serial number verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 1  
ParamName-1 = SerialNumberRange  
ParamString-1 = 100
```

FeatureID 1803 - Serial number duplicate verification for external accounts

To process the features for "Check serial number verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1803] - Parameter specification for serial number verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1804 - Amount range verification for external accounts

To process the features for "Check amount verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1804] - Parameter specification for amount range verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```
ParamCount = 2  
ParamName-1 = AmountRange  
ParamString-1 = 100  
ParamName-2 = MinStatisticalCycles  
ParamString-2 = 100
```

FeatureID 1805 - Check velocity verification for external accounts

To process the features for "Check velocity verification" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1805] - Parameter specification for check velocity verification

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

```

ParamCount = 4
ParamName-1 = MinStatisticalCycles
ParamString-1 = 30 ;e.g. 30 days
ParamName-2 = DiffMaxValue
ParamString-2 = 5 ; 5 %
ParamName-3 = DiffAverageValue
ParamString-3 = 4 ; 4 %
ParamName-4 = DiffCalAverageValue
ParamString-4 = 5 ; 5 %

```

FeatureID 1806 - Check amount highlighting for external accounts

To process the features for "Check amount highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1806] - Parameter specification for amount highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1807 - Account number highlighting for external accounts

To process the features for "Account number highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1807] - Parameter specification for account number highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

FeatureID 1808 - Branch/transit code highlighting for external accounts

To process the features for "Branch/Transit code highlighting" it is necessary that check data and statistical data (from today and from past) are present.

[FeatureParm-1808] - Parameter specification for branch/transit code highlighting

The feature parameters of the engine can only be defined in this section.

Note The parameters in this section CANNOT be defined in the [Common] section.

No feature parameters are required.

Example of a GIA SAE configuration

This is an example of an SAE use case with GIA:

- The workflow puts the check to queue 26
- The GIA server reads the check from queue 26 and sends it to the GIA client
- The GIA client puts it to the SAE engine
- In the SAE engine the FeatureIDs 1793, 1794 and 1795 are processed. It checks if the check serial number is in the defined ranges.

```
[GIA]
Register4Notification = Yes
Port4Notification = 2014
Queue4Notification = 26
Wait4Notification = 2

FeatureIDs = 1793,1794,1795
LoadRef2Engine = 1

[FeatureParam-1793]
ParamCount = 1
ParamName-1 = SerialNumberRange
ParamString-1 = 100 ; 100 increases the scope of
                    validity
[FeatureParam-1794]
ParamCount = 1
ParamName-1 = SerialNumberRange
ParamString-1 = 100 ; 100 increases the scope of
                    validity
```

ARV configuration example

Automatic Rules Verification

The ARV engine is verifying the signing rules for visually identified signature(s) automatically. The engine will take the IDs of the visually selected signatures and validate the rules according to the combined ASV process.

The configuration settings for an ARV Client and its corresponding server are to be set as follows:

In central configuration e.g. SrvMngr4.ini:

```
[ASVCL-X]
Program=automat2.exe ASV
Group=ASV
Host=
Port=2002
;The specific configuration file stored in the Configuration Server
  for an ARV client
RemoteIni=automat2arv.ini
ShowMode=0
Timeout=30
Start=No

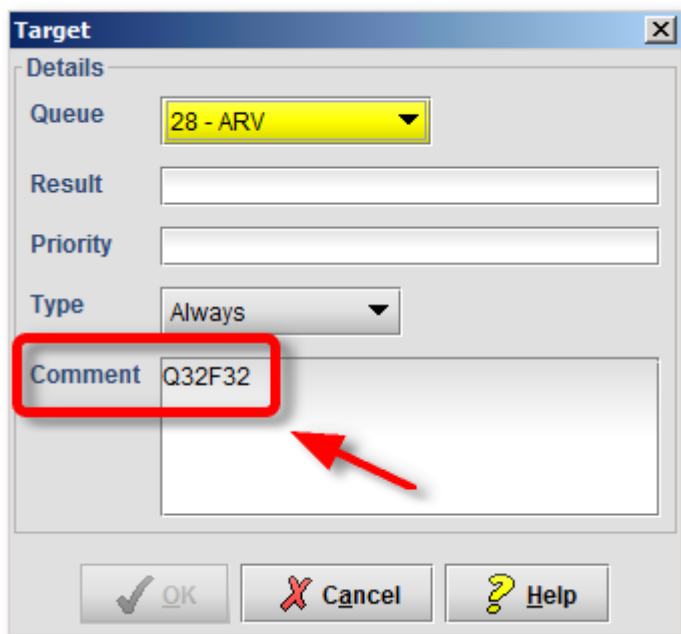
...
```

```
[ASV-Server-X]
Program=scs2.exe
Group=Automat
ShowMode=0
SignBase=W
Extensions=0126
SignCheck=WorkFlow
NoHistory=No
MsgTypeList=8A,8C
SCSnippetPrio=M
Sca2Ex=N
Latin=1,7
;The specific configuration of featureID for ARV and the queue that
;is used
ARVResultFeatureID=28 ; the result of the ARV is written with this
;FeatureID
ARVQueue=28 ; the queue with the selected signature
;number(s)
```

In Automat2arv.ini:

```
[Automat]
UseRestInst=Yes
CheckCodes=01, 02, 11
GiroCodes=17, 18, 19, 20, 21, 22, 23, 24, 50, 51, 52, 53, 54, 73, 76
IntlCodes=15
TwinTestRating=D5
;The specific configuration to turn on ARV and the queue that is used
;for notification
AutoRuleVerification=1 ; activate the ARV
Queue4Notification=28 ; queue for notification
```

Specific CRS routing target configuration in the Administration Client to indicate the source workflow items for Automatic Rules Verification. Indicated is the queue number and the feature ID where to find in the case of acceptance the ID(s) of the matching signature(s). They usually are coming out of the queues defined for the new VSOV display in the Java Client (e.g. VSV2 – Q32F32).



The image shows a 'Target' dialog box with a 'Details' tab. It contains several fields: 'Queue' is a dropdown menu showing '28 - ARV'; 'Result' and 'Priority' are empty text boxes; 'Type' is a dropdown menu showing 'Always'; 'Comment' is a text box containing 'Q32F32', which is highlighted with a red rectangle. A red arrow points to the 'Comment' field. At the bottom are three buttons: 'OK' (with a checkmark icon), 'Cancel' (with a red X icon), and 'Help' (with a question mark icon).

Figure 33: Routing target configuration for ARV

Important The queue 'Q' and the feature ID 'F' have to be set correctly in the comment field.