Informatica® Identity Resolution
10.1 HotFix 1

Operations Guide

Informatica Identity Resolution Operations Guide
10.1 HotFix 1
October 2019

© Copyright Informatica LLC 1999, 2019

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. Other company and product names may be trade names or trademarks of their respective owners.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2019-10-27

Table of Contents

Preface 8

Learning About Informatica Identity Resolution 8

What Do I Read If 9

Informatica Resources 9

Informatica Network 9

Informatica Knowledge Base 10

Informatica Documentation 10

Informatica Product Availability Matrices 10

Informatica Velocity 10

Informatica Marketplace 10

Informatica Global Customer Support 10

Chapter 1: Introduction 12

Overview 12

Conventions 12

Rulebase and Database Names 12

Database Object Names 16

Error Logs 18

Utility Locking 19

Chapter 2: Servers 20

Concepts 20

Configuration 22

Starting the Identity Resolution Servers 23

Default Configuration 24

Custom Configuration 24

Search and Rulebase Servers 25

Connection Server 27

Console Server 27

Stopping 29

Restarting 30

Server Statistics 31

Rulebase Server Groups 32

High Availability with Apache ZooKeeper 36

Configuring the ZooKeeper Properties 36

Starting and Stopping a ZooKeeper Server 37

Environment Variables 38

Windows Services 42

idssvc Utility 42

updsvc Utility 43

Chapter 3: Console Client . . . 45
Overview . . . 45
Starting . . . 45
Modes . . . 47
Window Layout . . . 48
Menu Items . . . 50
Starting from the Console . . . 55
Jobs Menu . . . 56
System Editor . . . 57
Log Viewer . . . 57
Chapter 4: Search Clients . . . 59
Overview . . . 59
Deployable Search Clients . . . 60
Administrator Search Clients . . . 61
Default Client . . . 63
Lite Client . . . 63
HTTP Search Client . . . 64
Operation . . . 65
Relate . . . 66
Starting from the Console . . . 66
Starting from the Command Line . . . 66
Report Formats . . . 70
Threads . . . 71
SQL Input . . . 71
XML Input . . . 71
Delimited Input . . . 72
DupFinder Mode . . . 73
Output View Layout . . . 73
DupFinder . . . 74
Chapter 5: Table Loader . . . 75
Concepts . . . 75
Starting . . . 75
Restarting . . . 77
Performance . . . 78
Sort Buffers . . . 79
Fault Tolerance - Data Errors . . . 80
Locales . . . 81
Chapter 6: Update Synchronizer . . . 83
Overview . . . 83

updsync utility. . . . 86
updmulti utility. . . . 90
Restarting Automatically. . . . 94
Synchronization Level. . . . 94
Transaction File / Table. . . . 96
Integrity Checking. . . . 99
Performance. . . . 99
Timing Window. . . . 101

Chapter 7: Globalization. . . . 102

Overview. . . . 102
Character Sets. . . . 102
Database Support for UNICODE. . . . 103
Binary Mode Utilities. . . . 105
Loading IDTs. . . . 106
IIR Clients. . . . 107
- Relate. . . . 107
- Java Search Client. . . . 108
- Synchronizer. . . . 108
- SSA-NAME3 V2. . . . 108

Debugging a Search. . . . 108
Miscellaneous Tips. . . . 109

Chapter 8: Siebel Connector. . . . 111

Overview. . . . 111
Configuring Siebel. . . . 111
- Constructing Load Data. . . . 112
- Synchronization Setup. . . . 112
- Integration Object. . . . 112
- IIR Business Service. . . . 113
- Error Handling. . . . 113
- Workflows. . . . 113
- Load Action Set. . . . 115
- Synchronization Action Sets. . . . 116
- Synchronization Run Time Events. . . . 117
- Profile Attributes. . . . 117

Configuring IIR. . . . 118
- System Definition. . . . 118
- Environment Variables. . . . 118
- Loading Data. . . . 119
- Synchronization. . . . 119
- XS Server. . . . 119
- Restrictions. . . . 120

Chapter 9: Web Services 121
Introduction 121
IIR Web Services 121
XML Search Service 122
XML Console Service 132
NSA-Batch Service 147
Real Time Web Service 148
Configuration Settings 149
Generic Mode 150
Custom Mode 150
Sequence Numbers 150
Operation Types 151
Real Time Reject table layout 151
Real Time Failure on System Refresh and Delete 151
Deploying a Real-Time Web Service 152
Example Soap Messages 153
Custom HTTP Header 157
UDDI 158

Chapter 10: ASM Workbench 161
Introduction 161
Launching the ASM Workbench 161
ASM Workbench Input Options 162
Country Specific Input 162
Character Set 163
Country Preload Option 163
Address Input Type 163
Options 164
Parsing and Validation Frame 164
Attributes 164
Suggested Address Label Display 165
Address Result Panel Display 166
Validation Status and Database Version Display 166
Output Result Frame Column Selection Menu 168
Field Status Display 169
CASS Field Status Display 171
CASS Summary Report Display 172
Statistics Reports - CASS Certification 173
File Menu Options 173
ASM Workbench and Batch Test utility 174

Chapter 11: System Backup and Restore. 176

Overview. 176

Back Up the System. 176

Restore the System. 177

Chapter 12: Batch Utilities. 178

Batch Utilities. 178

Common Parameters. 179

ssachdb Utility. 180

Synchronization Considerations. 180

dbinit Utility. 181

idsinit Utility. 181

lockmgr Utility. 181

version Utility. 182

idsbatch Utility. 182

iirconfig-tool Utility. 183

loggrabr Utility. 185

logfrmat Utility. 186

db_util Utility. 187

Command File Syntax. 187

Index. 190

Preface

Read the *Informatica Identity Resolution Operations Guide* to learn about the operation of the run-time components of Identity Resolution, such as servers, search clients and other utilities.

Learning About Informatica Identity Resolution

This section provides details of documentation available with the Informatica Identity Resolution product.

Introduction Guide

Introduces Identity Resolution and it's related terminology. It may be read by anyone with no prior knowledge of the product who requires a general overview of Identity Resolution.

Installation Guide

This manual is intended to be the first technical material a new user reads before installing the Identity Resolution software, regardless of the platform or environment.

Design Guide

This is a guide that describes the steps needed to design, define and load an Identity Resolution "System".

Developer Guide

This manual describes how to develop a custom search client application using the Identity Resolution API.

Operations Guide

This manual describes the operation of the run-time components of Identity Resolution, such as servers, search clients and other utilties.

Populations and Controls Guide

This manual describes SSA-Name3 populations and the controls they support. The latter are added to the Controls statement used within an IDX-Definition or Search-Definition section of the SDF.

Release Notes

The Release Notes contain information about what's new in this version of Identity Resolution. It is also summarizes any documentation updates as they are published.

What Do I Read If. . .

I am. . .

. . . a business manager

The INTRODUCTION to Identity Resolution will address questions such as "Why have we got Identity Resolution?", "What does Identity Resolution do"?

I am. . .

. . . installing the product?

Before attempting to install IIR you should read the INSTALLATION GUIDE to learn about the prerequisites and to help you plan the installation and implementation of the Identity Resolution.

I am. . .

...an Analyst or Application Programmer?

A high-level overview is provided specifically for Application Programmers in the INTRODUCTION to Identity Resolution.

When designing and developing the application programs, refer to the DEVELOPER GUIDE which describes a typical application process flow and API parameters. Working example programs that illustrate the calls to IIR in various languages are available under the `<IIR_client_installation>/samples` directory.

I am. . .

...designing and administering Systems?

The process of designing, defining and creating Systems is described in the DESIGN GUIDE. Administering the servers and utilities is described in the OPERATIONS manual.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction

This chapter includes the following topics:

- Overview, 12
- Conventions, 12

Overview

This manual describes the operation of the run-time components of Informatica Identity Resolution.

The components covered are as follows:

- Console Server and Client
- Search Server and Connection Server
- Update Synchronizer
- Online Rulebase Search Client and Applet
- Batch Rulebase Search Client (Relate and DupFinder)
- Batch Utilities
- Debugging facilities

The Rulebase editor is covered in the *Informatica SSA-NAME3 Application and Database Design Guide*.

Conventions

Identity Resolution uses the following naming conventions to manage the database objects:

Rulebase and Database Names

Identity Resolution Rulebase contains the rules that describe Systems. An Identity Resolution Database is the implementation of those rules, and the database contains IDTs and IDXs.

The Rulebase and Database are physically implemented as a set of tables and indexes in a relation database.

Identity Resolution supports multiple Database Management Systems, and the Rulebase and Database names consist of the following components:

- Database interface to use

- Interface-specific information

The name uses the following syntax:

```
Interface:Interface_specific
```

`Interface` identifies the database interface to access the DBMS. The supported values are as follows:

- `odb`. Specifies the ODBC interface. Supported target database types are Oracle, UDB/DB2, and Microsoft SQL Server. You can also use other ODBC data sources for unsynchronized source access.
- `ids`. Specifies the Dictionary Alias. You can use `ssa` instead of `ids`.

The format of the `Interface_specific` information is described in the following sections:

odb: Interface - ODBC

The `Interface_specific` information uses the following syntax:

```
SystemQualifier:Userid/Password@Service
```

The `Interface_specific` information uses the following values:

SystemQualifier

A number between 0 and 99, which qualifies the names of any database objects that Identity Resolution creates.

The default `SystemQualifiers` for the Rulebase and Database are 0 and 1 respectively. They must be different. For information about the naming conventions, see the *Database Object Names* section.

Userid

User name to access the database.

Password

Password for the user name.

Service

Name of the service that you specify in the `odbc.ini` file.

For information about ODBC configuration, see *Informatica Identity Resolution Installation and Configuration Guide*.

For example, `odb:0:scott/tiger@server` specifies an ODBC host DBMS. Identity Resolution prefixes the tables that it creates with `IDS_00_`(where the _00_ component is the `SystemQualifier`). Identity Resolution connects to the database identified as "server" using the user id "scott" and a password of "tiger."

Oracle Operating System Authentication

Identity Resolution supports Oracle's Operating System Authentication. In this scenario, clients can omit the Userid, Password, or Service when connecting to the servers. As the Identity Resolution processes initiate all database connections, they connect to Oracle using the operating system account ID of the user who launched them. Therefore, a user that has been granted access to Oracle must launch the servers.

For example, suppose the Identity Resolution Administrator's userid is SSA. Oracle is configured with the following parameters:

```
OS_AUTHENT_PREFIX = OPS$
REMOTE_OS_AUTHENT = TRUE (only if a Service is specified)
An
```

An Oracle Userid `OPS$SSA` is created with the appropriate privileges required by Identity Resolution. When a client specifies a Rulebase name of `odb:0:/@server734` and the host/port number of Rulebase Server, the

server connects to Oracle using the Administrator's userid and password. All database objects created by the server are in the `OPS$SSA` schema.

ids: Interface - Dictionary Alias

You can use alias name for the rulebase, database, or source name, which hides the actual connection string from the application programs.

Note: On UNIX platforms, only the Identity Resolution Administrator must have read and write permission on this file.

To use the alias feature, perform the following steps:

1. Create a text file that contains the alias names followed by their actual connection string, separated with tabs or spaces. For example:

   ```
   rb odb:0:username/password@dbserver
   db odb:1:username/password@dbserver
   src odb:99:username/password@dbserver
   ```

2. Define an environment variable in the server's environment to find the dictionary file.
   - On Windows: `set SSA_DBDICT=%SSAWORKDIR%\mydict.dic`.
   - On UNIX:

     ```
     SSA_DBDICT=$SSAWORKDIR/mydict.dic
     export SSA_DBDICT
     ```

   If you do not define this variable, the default dictionary file is `$SSABIN/dbdict.dic`.

3. Use the alias names instead of the actual connection string. You must set `Interface` to `ids` to enable this alias lookup feature. For example, use `ids:rb` to refer the rulebase, `ids:db` to refer the database and `ids:src` to refer the source.

ids: Interface - Encrypted Dictionary Alias

Identity Resolution can use an encrypted dictionary file to hide the actual connection string from the users who have access to the file system. The encrypted dictionary file is a text file that you can transfer through FTP in ASCII mode if needed.

Note: On UNIX platforms, only an Identity Resolution administrator must have the read and write permissions on the file.

Use the `iirdict` utility to create an encrypted dictionary file. The utility uses the 256-bit Advanced Encryption Standard (AES) algorithm for encryption.

You can specify the name of a file that you create or update as a command line argument. If you do not specify the name as a command line argument, the utility uses the file specified in the SSA_DBDICT environment variable. If you do not set the SSA_DBDICT environment variable, the utility uses the `$SSABIN/dbdict.dic` file.

```
$SSABIN/iirdict xxx.dic
iirdict <revision>
Enter a password:
Re-enter password:
Operating on 'xxx.dic'
Command (a=Add d=Delete l=List t=Test q=Quit)?
```

If the encrypted dictionary file does not exist, the utility creates the file and prompts you to provide a password. The password does not echo at the command prompt. If the encrypted dictionary file exists, you must enter the correct password.

You can use the following commands:

- a. Adds an entry to the encrypted dictionary.
- d. Deletes an entry from the encrypted dictionary.
- l. Lists the log of changes made to the encrypted dictionary.
- t. Tests a database connection to see if it is working.
- q. Exits the `iirdict` utility.

The following sample session adds an alias:

```
iirdict <revision>
Enter password:
Operating on 'xxx.dic'
Command (a=Add d=Delete l=List t=Test q=Quit)? a
Enter alias: rb
Enter connection details:
Type (odb):
System Qualifier: 1
Userid (ssa):
Password:
Re-enter password:
Service: dbserver
Connection string is 'odb:1:ssa/@dbserver'
(p=Proceed r=Re-enter d=Discard): p
iirdict> alias 'rb' tested successfully
iirdict> alias 'rb' added successfully
```

Note: If you update the dictionary file, ensure that you restart the Identity Resolution servers to reflect the changes.

You can also use the HTTP authentication mechanism to encrypt the connection string. To use the HTTP authentication mechanism, specify `http` in the `Type` prompt. The following sample session uses the HTTP authentication mechanism:

```
iirdict <revision>
Operating on 'D:\builds\a3pi\bin\dbdict.dic'
Enter password:
Command (a=Add d=Delete l=List t=Test q=Quit)? a
Enter alias: realm
Enter connection details:
  Type (odb): http
  Realm userid: user2
  Realm password:
iirdict> alias 'realm' added successfully
Command (a=Add d=Delete l=List t=Test q=Quit)? d
Comment: q
```

For information about these fields, see the Database Object Names section.

Note: Password does not echo at the command prompt.

When you enter an alias, the `iirdict` utility tries to validate the connection. The alias is added regardless of any connection errors, which can be caused by an external problem, such as an incorrect ODBC or database configuration.

For example, if you want to change the connection to change the password, you must delete the connection and add it again. The `iirdict` utility does not allow the file to contain two aliases with the same name.

Use the list command to list the log of changes made to the encrypted dictionary.

```
Command (a=Add d=Delete l=List t=Test q=Quit)? l
# Tue Feb 9 23:22:07 2010 ssa Created
# Wed Feb 10 00:09:29 2010 ssa Added alias 'rb'
```

In this example, ssa is the name of the Identity Resolution administrator.

You can define the SSA_DBDICT environment variable in the server's environment, as in the previous section. It defaults to `$SSABIN/dbdict.dic`.

Database Object Names

This section describes the way in which names are generated for IIR objects.

Control Objects

The following objects are created on the IIR Database (target database) to provide control information:

Object	Object Type
IDS_FDT_META	Table
IDS_FDT_META_DBID_I	Index
IDS_FDT_META_ID_I	Index
IDS_FDT_META_NAME_I	Index
IDS_FDT_META_NMIDDB_I	Index
IDS_FDT_RECID	Table
IDS_FDT_RECID_NO_I	Index
IDS_RB_GROUPS	Table
IDS_RB_GROUPS_I	Index
IDS_2PC	Table
IDS_UPD_SYNC_NSA	Table
IDS_UPD_SYNC_NSA_I	Index

ID Tables and Indexes

The following objects are created in the Identity Resolution database when an Identity Table is loaded. The object names include a prefix, `IDS`, and a two-digit system qualifier, `nn`. System qualifier is the numeral value in a database connection string. For example, if the database connection string is `odb:1:userid/password@service`, the system qualifier is 1.

Object	Object Type	Description
IDS_nn_IDTName	Table	ID Table
IDS_nn_IDLName	Table	Link Table
IDS_nn_IDTName_I[1..n]	Index	ID Table Indexes. _I is the RecId index. _I{1..n} are the PK / join indexes.

Object	Object Type	Description
IDSX_nn_IDXName	Table	IDX Table
IDSX_nn_IDXName_I	Index	IDX Index

where

`nn` indicates the system qualifier.

`IDTName` is the value of the NAME= parameter from the IDT-Definition.

`IDLName` is the value of the IDL-NAME= parameter from the `Multi-Search-Definition`.

`IDXName` is the value of the ID= parameter from the `IDX-Definition`.

RuleBase Objects

Object	Object Type
IDS_nn_INUSE	Table - Rulebase Lock
IDS_nn_SSARBN	Table - Data
IDS_nn_RECOVERY	Table - Restart/Recovery
IDSX_nn_SSARBN	Index

where

`nn` indicates the rulebase qualifier, which is the numeral value in a rulebase connection string. For example, if the rulebase connection string is `odb:0:userid/password@service`, the rulebase qualifier is 0.

Synchronizer Objects

The Update Synchronizer is supported by various objects which are created by the SQL scripts `updsyncu.sql` and `updsynci.sql`. These objects are created in the source database (containing User Source Tables).

Object	Object Type
IDS_UPD_SYNC_TXN	Table
IDS_UPD_SYNC_TXN_I	Index
IDS_UPDATE_SYNC_SEQ	Sequence
IDS_UPDATE_SYNC	Package
IDS_UPDATE_SYNC	Package Body
IDS `nnnnn`	Triggers

The System Loader will create triggers on the USTs if the SYNC option was specified. All triggers use the following naming convention:

- `IDS` fixed prefix

- `nnnnn` unique identifier (base 36 number)

Error Logs

IIR error logs may be output by various utilities and/or returned in response to an **ids_error_get** API call.

This is a sample Error Log created by the Table Loader:

```
ErrorLog: [ 1.773 2] loadit > It is now 20020612123407
ErrorLog: [ 1.773 2] exit code -252010410
ErrorLog: [ 1.543 2] loadit.c 3013 rc 10 32520104*100
ErrorLog: [ 1.493 2] thread_init failed
ErrorLog: [ 1.442 2] loadit.c 2494 rc 4 325201*100
ErrorLog: [ 1.392 2] match_rb_open returned -325201
ErrorLog: [ 1.342 2] utils.c 1342 rc 1 3252*100
ErrorLog: [ 1.292 2] connect to rb server failed -3252
ErrorLog: [ 1.242 2] sockapi.c 394 rc 2 23*10
ErrorLog: [ 1.192 2] socket.c 1765 rc 3 2*10
ErrorLog: [ 1.142 2] socket.c 581 rc 2
ErrorLog: [ 1.092 2] send(568) failed -1: winsock error 0
ErrorLog: [ 1.022 2] sockapi.c 627 rc 2 325*10
ErrorLog: [ 0.972 2] sockapi.c 595 rc 5 32*10
ErrorLog: [ 0.922 2] socket.c 1860 rc 2 3*10
ErrorLog: [ 0.872 2] socket.c 1833 rc 3
ErrorLog: [ 0.822 2] ssasocket_recv_n: received zero bytes
ErrorLog: [ 0.581 2] loadit.c 2848 rc 3 3691524*100
ErrorLog: [ 0.531 2] process_input: thread_init failed for thread
#1
ErrorLog: [ 0.481 2] loadit.c 2729 rc 24 36915*100
ErrorLog: [ 0.431 2] loadit.c 1448 rc 15 369*100
ErrorLog: [ 0.361 2] ssaxld_init failed -369:
ErrorLog: [ 0.311 2] dbops.c 4815 rc 9 36*10
ErrorLog: [ 0.201 2] ssaodbc.c 15352 rc 36
ErrorLog: [ 0.151 2] ssadb6_ssaxld_init failed: SSAST Could not
get memory
ErrorLog: [ 0.101 2] sort.c 1775 rc 13
ErrorLog: [ 0.050 2] sort.c 1305 rc 6
```

Interpreting an Error Log

Find the oldest message. It indicates the first error that occurred. It is identified by the smallest relative timestamp. For example, the last line from the log above is: `ErrorLog: [ 0.050 2] sort.c 1305 rc 6`

This line of data has the following format:

- Timestamp - message relative (0.050)
- Thread number (2)
- Module name (sort.c)
- Line number (1305)
- Response code (6)

We can infer that an error occurred in a sort routine. Continue up the stack in order of increasing timestamp looking for a text message. The messages containing module names and line numbers can be ignored. They simply give context information (a stack trace) of who called the function that reported the error.

The first text message is as follows:

```
ErrorLog:[0.151 2] ssadb6_ssaxld_init failed: SSAST Could not get memory
```

This message indicates that the sort routine failed when attempting to allocate some memory. In response to this you could add some RAM and/or increase the available swap space, or decrease the amount of memory required by the Table Loader by adjusting its parameters.

You could continue up the stack looking for more information. However, the first message is the important one. The other messages may report consequential errors, and are of less interest.

Some Error Logs will contain two error stacks. This typically occurs when two communicating processes fail. For example, when a search client (say relate) calls the Search Server which subsequently reports an error, both processes will report their Error Logs.

Utility Locking

Some utilities require exclusive use of certain IIR system resources. These include the:

- Table Loader
- Update Synchronizer
- Refresh / Delete utility

When the utility starts it will acquire application level locks within the Rulebase for the appropriate resources. Other processes that require the same locks will not be allowed to run.

For example, locks are used to prevent two Update Synchronizers from updating the same IDT and IDXs concurrently.

The `lockmgr` utility (documented in the *Batch Utilities* chapter) is used to list and delete locks in the situation when a utility terminates abnormally while holding locks. In most situations IIR is able to determine that the utility has crashed to unlock the resources automatically. The `lockmgr` utility can be used to manually unlock resources in the rare circumstances when automatic unlock is not possible.

System Name

Select a system from the list of available systems in the current Rulebase, to view the list of job names belonging to the system.

Job Name

Select a job name from the list of available jobs in the selected system, to view the run information.

System Logs

Check this option to see the System dependant logs.

Global Logs

Check this option to see the System independent logs.

Run-Information

The user is presented with a run information list of the selected system job. The user can at this time make a selection to view the relative step information.

Step Logs

The user is presented with a list of steps belonging to the selected job. The user can now view the run logs, error logs, output files (if any) of each step by selecting the desired option.

CHAPTER 2

Servers

This chapter includes the following topics:

- Concepts, 20
- Configuration, 22
- Starting the Identity Resolution Servers, 23
- Stopping, 29
- Restarting, 30
- Server Statistics, 31
- Rulebase Server Groups, 32
- High Availability with Apache ZooKeeper, 36
- Environment Variables, 38
- Windows Services, 42

Concepts

The following sections provide an overview of the concepts that are relevant to Identity Resolution.

Search Server

The Identity Resolution supports multiuser search and matching facilities by using the data stored in the IIR Tables. Search clients access the Search Server using an Application Program Interface (API).

The Identity Resolution Search Server is a multithreaded application, with one thread allocated to each client connection. Each Search Server process supports a limited number of connections, which are database and environment dependent. Multiple Search Server processes can be started to handle as many concurrent Search Clients as required.

Clients communicate with the server by using TCP/IP sockets. Ideally, the server is started on the same machine as the DBMS to avoid excessive network overhead when it communicates with the database.

The Search Server by default uses all available CPUs to provide the fastest possible matching. See Switches to see how to control how many match threads to use.

The Search Server maintains a pool of previously run search requests. The Search Server loads each search request into memory and initializes it. The Search Server uses the ids_search_start function to initialize a search request. When an ids_search_start function fails because of an transient error, the Search Server retries up to five times. The function does not retry when it returns a fatal error. If the function fails after five times, it returns a fatal error.

When the search request is complete, the Search Server adds the search request to the pool of search requests. The Search Server reuses these search requests, instead of reinitializing them whenever a client switches to another search. This method reduces overhead such as reading metadata and establishing database connections and improves the search performance.

By default, session pooling is enabled. You can disable it by setting the SSA_SESSION_POOL_MAX environment variable to 0 on the machine hosting the Search Server.

Rulebase Server

The Identity Resolution Rulebase Server supports multiuser access to the rules stored in the Rulebase. Search clients do not directly access this server. The Search Server, Console Server, and IIR utilities access the Rulebase Server.

The Rulebase Server caches rules read from the Rulebase to speed up access. One Rulebase server is permitted for each Rulebase (to maintain cache consistency). A single Rulebase Server can serve multiple clients and multiple Rulebases.

Connection Server

The Identity Resolution `Connection Server` is an optional server process. It is used to improve the performance of search clients that continually connect and disconnect from the Search Server.

Stateless transaction based searches, such as Web searches would benefit from using the Connection Server. For example, a Perl search client launched by a Web CGI-script might start a search, collect the results and terminate. Each search transaction opens a connection to the Search Server (and database) and closes it. This is inefficient.

To overcome this problem, use Identity Resolution to provide a `Connection Server`. When a transient search client connects to the Connection Server, the server allocates a `Session-Id`.

The Connection Server passes the request to the Search Server. The Connection Server returns the results to the search client without closing the connection to the Search Server. When the search client makes a second or subsequent call identified by the same `Session-Id`, the Connection Server reuses the connection established on the first call. It avoids the overhead of reconnecting with the Search Server. The connection is closed when the search client requests to terminate the session, or the session remain unused for the Connection Server's time-out period.

The client and Connection Server must reside on the same machine. It ensures that opening a (socket) connection from the client to the Connection Server is inexpensive (relative to connecting to a remote machine). If the client and Search Server use different character sets (example: `EBCDIC/ASCII`), the Connection Server must run on the same machine as the client. It is because the Connection Server does not perform any character set translations.

Console Server

The Identity Resolution Console Client accesses the Identity Resolution Console Server. Use the server to provide support facilities for the Console Client, such as RuleBase access, file access, and launching Identity Resolution utilities.

XML Console Server (CX)

The Identity Resolution XML Console Server is an optional server that implements the Console API using an XML protocol. For information about the XML Console Service, see the *Web Services* section of this guide.

XML Search Server (XM)

The Identity Resolution XML Search Server is an optional search server that implements the search API using an XML protocol. For information about the search API, see the *Developer Guide*.

Synchronization Server (XS)

The Synchronization Server is an optional server which has a Web Service-style interface to the following services:

- The Real Time Web Service, which propagates the User Source Tables updates to the IDT in real time.
- The Real Time API, which supports the client programs to apply updates to the IDT in real time.
- The NSA-Batch Service which integrates Identity Resolution with a Siebel CRM application using the Identity Resolution Siebel Connector. The Synchronization Server accepts XML messages from the Siebel Connector and stores them in the NSA Transaction Table.

HTTP Search Server

This optional server acts as a HTTP (Web) server to process search requests from a browser. Any browser can act as a web-based Search Client when pointed to the `host:port` of HTTP the Search Server . For information about configuring the HTTP Search Server, see the Search Clients chapter.

License Server

The License Server monitors a directory containing license files. These files define the products and optional components that might be installed and run in your environment. For information about the License Server, see the *Identity Resolution Installation Guide*.

Configuration

The following section discusses how to decide which servers are required and how to configure them. Server configuration and placement is important as it affects performance.

The Search and Rulebase Servers are packaged together in one executable image called `ssasrsv`. Based on start up parameters, `ssasrsv` can function as one of the following servers:

- Search Server
- Rulebase Server
- Search and Rulebase Servers

All clients communicate with the servers using a socket interface over TCP/IP. Communication costs vary based on where the client and server are placed. In order of most expensive to least expensive, they are:

- remote machines
- same machine
- same executable process

Clients on remote machines incur the cost of network transmission. Clients on the same machine will take a shortcut through the TCP/IP protocol stack and avoid the transmission costs. A combined Search and Rulebase server takes this one step further by bypassing TCP/IP altogether. Therefore it is advantageous to run a combined Search/Rulebase Server. If more Search Servers are required, they can be started as standalone Search Servers configured to access the Rulebase in the combined server.

Before launching the servers you must decide:

- which servers are required
- how many Search Servers are required
- which machines the servers will run on

The Console Server is required if you want to use the Console Client to administer Identity Resolution. A Connection Server is recommended for stateless search clients. At least, one search Server is required. In general, one Rulebase Server must be started.

Session Pooling Parameters

You can configure the session pooling method by setting the following environment variables in the `env\isss.bat` (Windows) or `env/isss` (UNIX) file:

SSA_SESSION_POOL_MAX

Maximum number of search requests that the Search Server can retain in the pool of search requests. The configuration does not impact the maximum number of search requests that the Search Server can run simultaneously. After the Search Server completes the search requests, it retains the maximum number of search requests that you had set in the pool.

Default is 100. You can disable the session pooling method by setting the SSA_SESSION_POOL_MAX variable to 0. Disabling the session pooling method results in the creation and destruction of the search-related resources based on the demand.

SSA_SESSION_POOL_TIMEOUT

Time period for a search request to remain unused in the pool of search requests. After the specified time period, the Search Server releases the unused searches from the pool, which frees the server resources and closes the database connections. You can specify the time period in seconds (s), minutes (m), or hours (h).

Default is 600 seconds, and the default unit is seconds. For example, if you set the value as SSA_SESSION_POOL_TIMEOUT=3600 or SSA_SESSION_POOL_TIMEOUT=1h, the Search Server releases the unused search requests from the pool after one hour.

SSA_SESSION_POOL_LOGGING

Level of logging that you want for a search request. The logging level ranges from 1 to 3, where 1 indicates minimal logging and 3 indicates extensive logging. Default is 0, which turns off the logging function.

Search Client Limit

The Search Server supports approximately 250 concurrent search clients. The limit is dependent on many factors, including available memory, free sockets, and the type of searches defined in your System. Multi-searches require more internal resources than individual searches, and searches that use SSA-NAME3 require more resources than searches using database indexes (such as Lite-Indexes). In the following discussion, the quoted limit of 250 must be understood as an approximate limit. Your implementation might support many more users, or in some particularly resource intensive cases, somewhat less than 250.

Will there be more than 250 concurrent search clients? If not, you must start a combined Search/Rulebase Server. If you expect more than 250 clients, start a combined Search/Rulebase Server, and one or more standalone Search Servers. It is assumed that your Application Server performs load balancing by distributing search client connection requests between all Search Servers.

Is your database in a cluster? If so, consider running multiple Search Servers and a Rulebase Server Group or multiple ZooKeeper Servers.

Starting the Identity Resolution Servers

Before you start the Identity Resolution Servers, ensure that the License Server is running.

Default Configuration

This section provides steps to start IIR server.

To start IIR Servers in the default configuration, click the **Start Server** icon in the Informatica program group (Win32), or run the shell script $SSABIN/idsup (UNIX platforms).

The default configuration starts a Console, Connection and combined Rulebase/Search Server. This configuration is suitable for most users. Errors encountered during startup are recorded in the server installation's `iirlog` directory.

Note: For Win32: The **Start Server** icon runs a script in the server installation's bin directory called `idsup.bat`.

For UNIX: Some platforms require the use of `nohup` when launching servers. Example, `nohup $SSABIN/idsup &`

Custom Configuration

If you wish to run servers in a custom configuration (such multiple Search Servers or with Rulebase Server Groups), you will need to write your own scripts to start and stop servers. The following section describes the parameters required to start individual servers.

Configure Mode (Install tests)

Identity Resolution Servers can be started in a special mode known as Configure Mode. This mode is used to start servers in the default configuration and run the standard installation test. When servers are started in this mode the first Console Client to connect to the server will automatically run the install test. Once the test has completed successfully the servers will automatically switch out of Configure Mode and behave as normal servers.

For Win32: Servers will be started automatically in **Configure Mode** by the installation process if you check the `Run Tests` checkbox. If the option is not selected during the installation phase they may be started later using the **Start Server (Configure Mode)** icon in the **Informatica** program group.

For more information about starting the server in the configure mode, see the *Informatica Identity Resolution Installation and Configuration Guide*.

Host Names / IP Addresses

Identity Resolution Server start-up parameters usually include a host name. Although not explicitly noted in the following parameter descriptions, an IP address may be substituted for a host name.

Sample Server Start-up and Shutdown Scripts

The Windows Identity Resolution Server installation contains two sample scripts, idsseup.bat and idssedn.bat, that can be used to start Server processes in various configurations.

Note: These scripts do not support Rulebase Server Groups.

To use these scripts you need to set the following environment variables:

SSA_SESV_RBPORT

Set to the port number that the Rulebase Server will be listening on. Set to 0 (zero) to prevent the Rulebase Server process from starting/stopping. In this case a separate Rulebase Server process must be running and the environment variable *SSA_SESV_RBHOST* must be set to the `host:server` address of the Rulebase Server process.

SSA_SESV_SEPORT

Set to the port number that the Search Server will be listening on. Set to 0 (zero) to prevent the Search Server process from starting/stopping.

SSA_SESV_XMPORT

Set to the port number that the XML Search Server will be listening on. Set to 0 (zero) to prevent the XML Search Server process from starting/stopping.

SSA_SESV_XSPORT

Set to the port number that the Synchronization Server will be listening on. Set to 0 (zero) to prevent the Synchronization Server process from starting.

SSA_SESV_HTPORT

Set to the port number that the HTTP Search Server will be listening on. Set to 0 (zero) to prevent the HTTP Search Server process from starting/stopping.

SSA_SESV_HOST

The `host` name of the computer on which the various server processes are running. This variable is used only by the `idssedn` script.

Search and Rulebase Servers

Specify appropriate switches to start an XML Search Server, Synchronization Search Server, HTTP Search Server, Rulebase Server, or a combined Search and Rulebase Servers.

Run the following command to start a Search Server or a Rulebase Server:

- On Windows: `%SSABIN%\ssasrsv <Switches>`
- On UNIX: `$SSABIN/ssasrsv <Switches>`

If you want to start a Search Server or a Rulebase Server as a background process, run the following command:

- On Windows: `start "<Window Title>" %SSABIN%\ssasrsv <Switches>`
- On UNIX: `$SSABIN/ssasrsv <Switches> &`

Use the following switches when you start a Search Server or a Rulebase Server:

-n<SePort>

Starts a Search Server that listens for client connections on the specified port number.

-x<XmPort>

Starts an XML Search Server that listens for client connections on the specified port number.

-s<XsPort>

Starts a Synchronization Server that listens for client connections on the specified port number.

-disable-idtlock

Removes the lock on the Identity Table. If the Identity Table is not locked, you can run multiple Synchronization Servers in parallel. When you have multiple transactions for a record, different Synchronization Servers might process the transactions out of order, which can cause data integrity errors. Use this switch with caution.

-H<HtPort>

Starts a HTTP Search Server that listens for client connections on the specified port number.

-m<RbPort>

Starts a RuleBase Server that listens for client connections on the specified port number.

-h<Host>:<Port>

Starts a Search Server configured to access a remote Rulebase Server. Specify the host name of the Search Server and the port number on which the server listens for client connections. Use the `-h` switch with the `-m` switch.

-readonly

Starts the Search Server and the Rulebase Server, and runs the Rulebase Server in the read-only mode.

The following sample command runs the Rulebase Server in the read-only mode:

```
$SSABIN/ssasrsv -readonly -mlocalhost:1666 -nlocalhost:1667 &
```

-readonly-no-socket

Starts the Search Server and the Rulebase Server, and runs the Rulebase Server in the read-only mode. Use this switch when you do not want to specify a port number for the Rulebase Server.

The following sample command runs the Rulebase Server in the read-only mode:

```
$SSABIN/ssasrsv -n$SSA_SEPORT -readonly-no-socket &
```

-g<Rulebase Server Group>

Specifies the Rulebase Server Group.

-use-abort

Applicable only for UNIX. Performs a forced restart of the Rulebase Server. Use this switch only if the Rulebase Server does not restart after a database connection failure.

-w<n>

Specifies the polling frequency of a Rulebase Server Group in seconds. Default is 1.

-z<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>

List of ZooKeeper servers based on the host name of the ZooKeeper server and the port on which the host listens for connections.

-z<n>

Specifies the number of requested match threads. The default value is 1. Specify this argument if your typical usage is a small number of long running searches that might require multiple threads.

-y<Max>[,<Wait>]

Specifies the maximum number of times to retry (Max) when a connection to the database fails and the number of seconds (Wait) that a Search Server or the Rulebase Server waits to retry the connection. For example, `-y5,3` indicates that the Search Server or the Rulebase Server can try up to five times to connect to the database for every three seconds. The default value is `0,0`. If `Max=0`, the Search Server or the Rulebase Server retries indefinitely until the connection to the database succeeds.

-1<File>

Specifies the file that logs the standard output messages (`stdout`).

-2<File>

Specifies the file that logs the standard error messages (`stderr`).

-3<File>

Optional. Specifies the file that logs the error and debug messages.

-u<Rulebase Options>

Controls optional aspects of the Rulebase Server behavior.

Use the following Rulebase Server options:

- 0x0001 (1_{10}). Stores the Rulebase cache in memory when no users are currently connected. Specifying this option improves the Rulebase Server performance, and omitting this option reduces the memory utilization.
- 0x0100 (256_{10}). Forces the Rulebase Server to restart when a Rulebase read operation fails. This option is helpful when the database server bounces and disconnects the Rulebase Server's connections. If you do not set this option, the Rulebase Server might fail a client's requests that require database access. However, the clients that access the cached rules function normally.

The value that you specify with the `-u` switch is treated as a decimal value, unless you prefix the value with `x`. You can specify a combination of options by adding the values together. For example, to store the Rulebase cache in memory and force the server to restart on a read error, specify `-ux101` or `-u257`.

Use the following guidelines when you start a server:

- To start a combined Search and Rulebase Servers, specify the `-n` and `-m` switches.
- To start a standalone Search Server, specify the `-n` and `-h` switch.
- To start a standalone Rulebase Server, specify the `-m` switch.
- To start a standalone XML Search Server, specify the `-x` switch.
- To start a standalone Synchronization Server, specify the `-s` switch.
- To start a standalone HTTP Search Server, specify the `-H` switch.

Connection Server

The IIR Connection Server can be started from the command line as follows:

For Win32: %SSABIN%\ssacosv Switches

For UNIX: $SSABIN/ssacosv Switches

where `Switches` are

`-hHostname:hostport` This is the hostname or IP address of the machine where the IIR Search Server is running. If not supplied, the IIR Search Server is assumed to be running on the same machine as the Connection Server. The `hostport` enables you to specify the port number used by the IIR Search Server.

`-nListenPort` Specifies the port number to use when listening for client connections. The default port number is 1667.

`-tTimeout` Specifies the timeout value for a session in seconds.

`-1File` Specifies the file where messages written to `stdout` will be redirected.

`-2File` Specifies the file where messages written to `stderr` will be redirected.

Console Server

The IIR Console Server can be started from the command line as follows. Optional servers are not started if their `host:port` is not specified (-h).

For Win32: %SSABIN%\ssacssv Switches

For UNIX: $SSABIN/ssacssv Switches

where `Switches` are

-nPort

Defines the Console Server's port number. If the default port number of 1669 is already used by another application, use this parameter to request a different value. Any client connecting to this server would then have to specify the same port number.

-hrbHost:Port

Rulebase Server's Host name and port.

-hseHost:Port

Search Server's Host name and port.

-hcoHost:Port

Connection Server's Host name and port.

-hhtHost:Port

Optional HTTP Server's Host name and port.

-hxmHost:Port

Optional XML Server's Host name and port.

-hxsHost:Port

Optional XS Server's Host name and port.

-1File

Specifies the file where messages written to stdout will be redirected.

-2File

Specifies the file where messages written to stderr will be redirected.

-3File

Specifies the file where error and debug messages will be written.

-wWorkDir

Specifies the working directory for the Console Server process.

-zn

Passes through as argument to Search Server.

-m

Specify Rulebase Server options to be passed to the spawned RB server. This option uses the same values as documented in the *RB Server Options* section but is prefixed by -m instead of -u.

-o

Launch the Console Server without launching the Connection and combined Rulebase/Search Servers.

-tDirectory

Specifies the absolute name of the directory which contains the test files used in Configure Mode. The install test is in `$SSAWORKDIR/systems`. On Win32 platforms this parameter is supplied by the Installer.

-iFile

Informs the server to start in Configure Mode. This option is set in order to complete a new installation. It causes the first console client to start a session to run the install test. File is a file in the directory specified by the -t switch. It contains a list of system import files. These files are used during the testing phase of the setup. The name of this file should be `tests.dat`. On Win32 platforms the server is started in this mode by the Installer.

-uUID -pPWD -sSVC

These specify the User's Database Userid, Password and Service to be used when communicating with the Database in Configure Mode. They are passed to the client as default values to be used during the test. If not supplied, these values default to blanks. If any of these options are supplied in "normal" mode they are ignored.

Stopping

This section provides information about how to stop the servers.

Default Configuration

Win32

Servers are stopped using the Server Shutdown icon in the Informatica's Products folder or by running the script %SSABIN%\idsdown.

UNIX

Server are stopped using the script $SSABIN/idsdown. It must be run from a shell that has the Informatica's environment variables set (by sourcing the `ssaset` script first).

Normal vs Hard Shutdown

Under normal circumstances, a server will shutdown when all active clients disconnect from it. In some cases it may be desirable to request an immediate shutdown, for example, when the stop request has come from a Windows Service just prior to O/S shutdown. In this case, `idsdown` may be called with the hard parameter, which forces an immediate shutdown by closing all active client connections.

Custom Configuration

Use the `ssashut` utility to stop individual servers or close (flush) sessions held by the Connection Server.

For Win32: %SSABIN%\ssashut Switches

For UNIX: $SSABIN/ssashut Switches

where `Switches` are

-hHost:Port

Host and Port specify the host name and port number of the server to be shut down.

-f

Flush sessions instead of shutting down the server.

-v

Verbosity.

-z

Hard shutdown. This option forces the server to shutdown immediately by closing active connections. Any active clients will receive socket-related error messages. This option is mutually exclusive with -f.

Note: `ssashut` may report that a connection could not be established to the nominated server. Some of the possible reasons include:

- Wrong host or port number or both was specified, or

- Server is not currently running.

Note: See also the description of the Windows sample script `idssedn.bat` in the *Sample Server Start-up* and *Shutdown Scripts* sections above.

Restarting

All servers (Connection Server, combined Search/Rulebase Servers and Console Server) are launched as a pair of processes. The first process spawns a second server process that acts as the real server that clients connect to. If the spawned server crashes, the parent process automatically spawns a new copy of itself. This provides a degree of fault tolerance.

Rulebase Server

The Rulebase Server has special restart requirements because it uses a locking mechanism to protect itself. The locking mechanism prevents two Rulebase Servers updating the same Rulebase tables.

When a parent Rulebase server starts, it generates a unique Id and passes it to the child server. When the child opens the Rulebase it saves the Id in the Rulebase.

If another Rulebase server attempts to open the same Rulebase, its Id will not match the value held in the Rulebase and an error message similar to this is displayed:

```
Rulebase is locked
Rulebase In use by ssa.identitysystems.com IP=203.2.203.105 on port=1668,
ID=271259152
IS ANOTHER RULEBASE SERVER RUNNING?
```

Automatic Restart

If the child server crashes, the parent server spawns a new child with the same Id as the original child.

When the child server starts and finds an Id already present, it compares it to the parent's. If they are the same it displays the following message and restarts successfully:

```
This is an automatic restart
```

Manual Restart

If the computer crashes (and all processes terminate), the Rulebase remains locked. The next time a new pair of parent/child Rulebase Servers are started, the parent generates a new unique Id. It will not match the Id stored in the Rulebase, so the child server will fail to start. In this situation, you can manually override the lock by setting an environment variable to the same Id that currently locks the Rulebase. For example,

```
SSA_RB_RESTART_ID=271259152
```

When the Rulebase server is started, it will use the environment variable to unlock the Rulebase (as long as the two Ids match). It will then use the freshly generated parent Id to re-lock it. Therefore the environment variable can only be used once to unlock the database. A manual restart generates the following message in the server log:

```
This is a manual restart
```

Automatically Restarting After Common Failures

A manual restart is usually required after a power outage or reboot. When `SSA_RB_RESTART_ID` is set to 0, IIR will automatically attempt to detect if the original process that locked the Rulebase is still running. If it is not, the restart will be automatic (with no intervention required).

```
IS ANOTHER RULEBASE SERVER RUNNING?
Rulebase 'sdb:file:c:\a3i\ids\rule' In use by
```

```
ssa.identitysystems.com IP=203.2.203.109 on port=1668, SSA_RB_RESTART_ID=281728582
Other RB information: ip=203.2.203.109 pid=299
host='ssa.identitysystems.com' ps='2002/11/28 06:29:10.8233'
Other RB server not running
This is an automatic restart
```

However, if the original job is running, or its status cannot be determined, IIR will not automatically unlock the Rulebase.

Note: When `SSA_RB_RESTART_ID` is set to 0 it is possible to inadvertently start multiple Rulebase servers. If this occurs, Rulebase corruption will result. We strongly recommend that `SSA_RB_RESTART_ID` is not left in any start-up script or in the environment after a server restart.

Note: This facility requires that various operating system functions will return consistent results. For example, the host name of the machine and the output of the ps command must return the same result when called repeatedly. Any inconsistencies may result in the Server concluding that the previous server is no longer running and to start a second instance. If the previous server is still running, Rulebase corruption may result.

Connection Aliases

Users must not connect to the same Rulebase Server using multiple userids or service names that are aliases for the same physical rulebase.

For example, if the service names Jupiter and Mars are aliases for the same Oracle service, all rulebase connection strings must specify either Jupiter (or Mars) but must not use a mixture. Similarly, when using Oracle's Operating System Authentication feature, a connection string that explicitly provides a userid may be an alias of one that does not, as they may both be routed to the same physical rulebase. In this case, use one consistent form for the connection string, or consider using a dictionary alias.

The rulebase name is used to identify a cache containing updated rulebase data. The use of alias names will create multiple caches, which will be written to the same physical rulebase, causing corruption.

IIR will detect if an alias is used and refuse to open the connection, stating that the rulebase is already owned by another user.

Server Statistics

Progress information can be retrieved for the servers, which are themselves jobs started by the Console. See *Console Client* below for details about progress information. The slider can be used to slow the refresh rate from once per second (the default) to up to 30 seconds.

Because this has the potential to impact performance, it is not switched on by default. Some environment variables are required to be set in order for this feature to become available.

In the `<Identity Resolution Installation Directory>/env/isss` script, set the SSA_SERVER_STATS environment variable to YES, and set the SSA_RBNAME environment variable to the rulebase connection string with the rulebase number that you currently use.

For example:

- On Windows: `set SSARB_NAME=odb:0:userid/password@service` or `set SSARB_NAME=iir:rb`
- On UNIX: `export SSARB_NAME="odb:0:userid/password@service"` or `export SSARB_NAME="iir:rb"`

Note: To keep your password secure, Informatica recommends that you use a Dictionary Alias.

When you start the servers, issue a refresh. The jobs window displays the search server progress information.

Note: If the rulebase has only just been created, first use the console client to stop and restart the servers.

There will be two entries. One will be an overview job whose function is to restart the servers if one fails. It will state how long it has been running and what servers are active. Its logs are often interesting though.

The other will have the progress details of the search servers, if `SSA_SERVER_STATS=YES`. Otherwise it will merely list the individual servers and their start times.

The progress will look something like this:

```
ssasrsv: server 0:28:14.000
rulebase server: active
clients 4
rulebases 1
status available
search server: active
== Search clients ====
formerly active clients: 6
currently active clients: 1
maximum concurrent clients: 2
minimum duration: 0.000 seconds
maximum duration: 30 minutes 28.979 seconds
total duration: 37 minutes 21.435 seconds
average duration: 320.205 seconds
==== Searches ====
formerly active clients: 53558
currently active clients: 0
maximum concurrent clients: 1
minimum duration: 0.004 seconds
maximum duration: 1.692 seconds
total duration: 3 minutes 58.877 seconds
average duration: 0.004 seconds
==== Name3 clients ====
formerly active clients: 2
currently active clients: 0
maximum concurrent clients: 2
minimum duration: 6 minutes 37.422 seconds
maximum duration: 6 minutes 37.532 seconds
total duration: 13 minutes 14.954 seconds
average duration: 397.477 seconds
```

A particular job may run a series of searches, some in parallel. The maximum and minimum duration are recorded rather than the average. Generally speaking, a large maximum that continues getting larger indicates a client that has failed to disconnect. It can be seen that a small number of search clients can carry out a large number of searches. The average can be found by dividing the total duration by the total number of searches. Here `37m21s = 2241s/7 = 320s`

Rulebase Server Groups

Use the Rulebase Server Groups (RBSG) to run Rulebase Servers on every node of a cluster.

Overview

Rulebase Server Groups provide Rulebase redundancy by allowing several Rulebase Servers to run concurrently. Only one Rulebase Server is permitted to respond to queries (known as the Primary).

The other Rulebase Servers (the Secondaries) periodically poll the Primary to determine if / when it is no longer working so that one of them may assume the role of the Primary.

This feature is designed to be used in a distributed database (cluster) environment where multiple database servers are running on different network nodes (machines) while presenting a unified appearance as a "single

database". All nodes are connected to a shared disk sub-system by a storage area network. From a database client's perspective, the database remains available when at least one of the nodes remains operational. Additional nodes can be started or shutdown transparently, without affecting database client connectivity or data integrity.

Different software vendors use different names for this sort of technology. For example, Microsoft SQLServer Clusters and Oracle's Real Application Clusters are capable of providing the database functionality and a Veritas Cluster Server could be used to provide the Operating System dependent cluster related services.

The RBSG implementation does not make use of any proprietary features from any particular vendor. It simply requires that a consistent view of the database remains accessible when at least one node is operational.

Discussion

The database instance that the Rulebase Server is accessing must be absolutely robust, in the sense that there is no possibility of it becoming unavailable while there is a working network and at least one working machine in that network.

Only one Rulebase Server will respond to Rulebase requests no matter how many Rulebase Servers are in the group.

A RBSG is shut down using the `idsdown` script. This will stop all Rulebase Servers. There is no mechanism to shut down an individual Secondary Rulebase Server other than to kill it.

The Search and Rulebase Servers must always be started as separate servers and communicate through sockets. A combined Search / Rulebase Server is not permitted.

A unique `SSA_RB_RESTART_ID` for a RBSG will be set once when the group starts and will remain unchanged for the life of the group.

The Console Server must not automatically launch any other servers. You must start it with the `-o` switch.

Use the `-g` switch to assign Rulebase Servers to groups.

Parameters for the Rulebase and Search Servers

When you start a Rulebase or Search Server, you can specify the following parameters:

-e

Indicates not to shut down the secondary servers when the primary server shuts down.

-g<Rulebase Server Group Connection Name>,<Rulebase Connection String>

Adds the server to the Rulebase Server Group.

Use one of the following formats for the rulebase connection string:

- `odb:0:userid/password@service`
- `iir:rb`

-o<Restart Option>

Indicates how the server must behave when the connection to the database is lost. You can use one of the following options:

- `r`. Indicates to restart the server.
- `1`. Indicates to check whether the database table is accessible. If the database table is accessible, the server connects to the database. If the database table is not accessible, the server waits until the database connection is restored.
- `2`. Indicates to restore the database connection.

- `0`. Indicates to retry until the database connection is restored.

-t<Number of Retries>,<Frequency>

Indicates the maximum number of attempts for the secondary server to establish the database connection and the time interval between two attempts. Default number of attempts is 500, and default time interval is 5 seconds.

-w<Frequency>,<Priority>

Indicates the polling frequency in seconds and the priority for the server. When the primary server is down, the secondary server with the highest priority becomes the primary server. The number 1 indicates the highest priority. Default polling frequency is 1.

-m<Port Number>

Indicates the port number on which the Rulebase Server listens.

-n<Port Number>

Indicates the port number on which the Search Server listens.

Parameters for the Console Server

When you start a Console Server, you can specify the following parameters:

-g<Rulebase Server Group Connection Name>,<Rulebase Connection String>

Adds the server to the Rulebase Server Group.

Use one of the following formats for the rulebase connection string:

- `odb:0:userid/password@service`
- `iir:rb`

-n<Port Number>

Indicates the port number on which the Console Server listens.

-o

Indicates not to start other servers.

-h<Host Name>

Indicates the server that you want to automatically start.

Parameters for the XML Search Server

When you start an XML Search Server, you can specify the `-x<Port Number>` parameter that indicates the port number on which the XML Search Server listens.

rbsgdown Utility

Use the rbsgdown utility to shut down all the primary and secondary Rulebase Servers. You can specify the command at any node. The `rbsgdown` utility stops all the clients connected to the Rulebase Servers. The `rbsgdown` utility cannot be used for other servers and is specifically for the Rulebase Servers.

The following is a code sample to specify the command:

```
rbsgdown -gfranky,<Rulebase Connection String>
```

Use one of the following formats for the rulebase connection string:

- `odb:0:userid/password@service`
- `iir:rb`

Example

The name of the RBSG used in this example is `franky`.

The environment variable *%SSA_GRPDB%* contains the connection string to the cluster database. This database must contain the Rulebase objects and the `IDS_RB_GROUP` table. For example, it might be defined as `odb:99:uid/pwd@clusterdb`.

Start the first Rulebase Server in the group:

```
set SSA_PRM="IIR rb1 Server for group port 9997"
set SSA_LOGS=-1%SSAWORKDIR%\idsrb1v.log -2%SSAWORKDIR%\idsrb1v.err -3%SSAWORKDIR%
\idsrb1v.dbg
set SSA_ISSUP_CMD=start %SSA_PRM% "%SSABIN%\ssasrsv"
%SSA_ISSUP_CMD% -m9997 -gfranky,%SSA_GRPDB% -w1 %SSA_LOGS%
```

UNIX example

```
SSA_LOGS="-1$SSAWORKDIR/idsrb1v.log -2$SSAWORKDIR/idsrb1v.err -3$SSAWORKDIR/idsrb1v.dbg"
export SSA_LOGS
$SSABIN/ssasrsv -m9997 -gfranky,$SSA_GRPDB -w1 $SSA_LOGS
```

Start a second Rulebase Server in the same group:

```
set SSA_PRM="IIR rb2 Server for group port 9999"
set SSA_LOGS=-1%SSAWORKDIR%\idsrb2v.log -2%SSAWORKDIR%\idsrb2v.err -3%SSAWORKDIR%
\idsrb2v.dbg
set SSA_ISSUP_CMD=start %SSA_PRM% "%SSABIN%\ssasrsv"
%SSA_ISSUP_CMD% -m9999 -gfranky,%SSA_GRPDB% -w1 %SSA_LOGS%
```

UNIX example

```
SSA_LOGS="-1$SSAWORKDIR/idsrb2v.log -2$SSAWORKDIR/idsrb2v.err -3$SSAWORKDIR/idsrb2v.dbg"
export SSA_LOGS
$SSABIN/ssasrsv -m9999 -gfranky,$SSA_GRPDB -w1 $SSA_LOGS &
```

If the two servers are started on the same machine they must have different port numbers (9997 and 9999 respectively). If they are started on different machines they could use the same port numbers.

We now have two Rulebase Servers running. One will become the Primary Rulebase for this RBSG and the other will go into Secondary polling mode where it will just monitor the first Rulebase and take over if it detects that the Primary Rulebase has ceased to work.

We may start as many Rulebase Servers as necessary. All additional servers will become secondary servers.

Start a Search Server:

```
set SSA_PRM="IIR se Server on %SSA_SEHOST%"
set SSA_LOGS=-1%SSAWORKDIR%\idssexx.log -2%SSAWORKDIR%\idssexx.err -3%SSAWORKDIR%
\idssexx.dbg
set SSA_ISSUP_CMD=start %SSA_PRM% "%SSABIN%\ssasrsv"
%SSA_ISSUP_CMD% -n%SSA_SEPORT% -gfranky,%SSA_GRPDB% %SSA_LOGS%
```

UNIX example

```
SSA_LOGS="-1$SSAWORKDIR/idssexx.log -2$SSAWORKDIR/idssexx.err -3$SSAWORKDIR/idssexx.dbg"
export SSA_LOGS
$SSABIN/ssasrsv -n$SSA_SEPORT -gfranky,$SSA_GRPDB $SSA_LOGS &
```

Do not assign a RB Server port to the Search Server, as it will automatically determine the correct one based on the -g parameter. An error will be generated if a RuleBase Server and the -g switch are both specified.

Start the Console Server:

```
set SSA_PRM="IIR cs Server on %SSA_CSHOST%"
set SSA_LOGS=-1%SSAWORKDIR%\idscsxx.log -2%SSAWORKDIR%\idscsxx.err -3%SSAWORKDIR%
\idscsxx.dbg
set SSA_ISSUP_CMD=start %SSA_PRM% "%SSABIN%\ssacssv"
set SSA_ISSUP_HOSTS=-hco%SSA_COHOST% -hse%SSA_SEHOST% -hxm%SSA_XMHOST% -gfranky,
```

```
%SSA_GRPDB% %SSA_ISSUP_CMD% -o -n%SSA_CSPORT% %SSA_ISSUP_HOSTS% -w%SSAWORKDIR%
%SSA_LOGS%
```

UNIX example

```
SSA_LOGS="-1$SSAWORKDIR/idscsxx.log -2$SSAWORKDIR/idscsxx.err -3$SSAWORKDIR/idscsxx.dbg"
export SSA_LOGS
SSA_ISSUP_HOSTS="-hco$SSA_COHOST -hse$SSA_SEHOST -hxm$SSA_XMHOST -gfranky,$SSA_GRPDB"
export SSA_ISSUP_HOSTS
$SSABIN/ssacssv -o -n$SSA_CSPORT $SSA_ISSUP_HOSTS -w$SSAWORKDIR $SSA_LOGS &
```

Do not assign a Rulebase Server port to the Console Server, as it will automatically determine the correct one based on the -g parameter. Use the -o switch to prevent Search and Rulebase Servers from being spawned automatically.

High Availability with Apache ZooKeeper

Identity Resolution uses Apache ZooKeeper to achieve high availability for Rulebase Servers, updsync utility, and updmulti utility in a distributed environment. High availability refers to the availability of the resources without any service interruption if a failure occurs.

When you start a Rulebase Server, it starts as a secondary server, and one of the secondary servers become the primary server. The primary server responds to all the queries. If the primary server is down, any secondary server is elected as the primary server, and the new primary server responds to the queries.

If you specify the local host as a ZooKeeper server during the installation, the installer installs ZooKeeper.

Configuring the ZooKeeper Properties

The installer creates a configuration file named `zoo.cfg` that contains the ZooKeeper properties. You can customize the default properties of ZooKeeper based on your environment and requirement.

For more information about the ZooKeeper properties, see the ZooKeeper documentation.

1. Open the `zoo.cfg` file from the following directory:
   - On Windows: `<Identity Resolution Installation Directory>\bin`
   - On UNIX. `<Identity Resolution Installation Directory>/bin`
2. Customize the following properties based on your requirement:

   tickTime

   Length of a single tick, in milliseconds. ZooKeeper uses tick as the basic time unit to regulate timeouts. Default is 2000.

   initLimit

   Number of ticks after which the ZooKeeper server times out during the synchronization process. Default is 10.

   syncLimit

   Maximum number of ticks for the followers to wait to synchronize with the leader before the followers time out. Default is 5.

dataDir

Absolute path for the ZooKeeper to store the in-memory snapshots of the data and the transactional log of updates to the database.

clientPort

Port on which the ZooKeeper server listens for client connections.

maxClientCnxns

Maximum number of concurrent connections that a single client can make to a single member of the ZooKeeper ensemble. Default is 10.

3. If you use an ensemble of ZooKeeper servers, perform the following tasks:
   a. Add the following server entries to the `zoo.cfg` file:

```
server.1=<Host Name1>:<Port Number1>:<Port Number2>
server.2=<Host Name1>:<Port Number1>:<Port Number2>
server.3=<Host Name1>:<Port Number1>:<Port Number2>
...
server.N=<Host Name1>:<Port Number1>:<Port Number2>
```

   `Host Name` refers to the host name of the ZooKeeper server, and `Port Numbers 1 and 2` refers to the ports that the ZooKeeper uses to communicate between the ZooKeeper servers.

   For example, the following entries indicate an ensemble of three ZooKeeper servers:

```
server.1=PriHost:18046:18047
server.2=SecHost:18049:18050
server.3=TerHost:18052:18053
```

   b. Create a file named `myid` in the directory that you configure for the `dataDir` property in the `zoo.cfg` file.

   c. Open the `myid` file, specify the ID of the server based on the `server.<Number>` entry in the `zoo.cfg` file, and save the `myid` file.

   For example, if the host name is specified as `server.2` in the `zoo.cfg` file, specify 2 in the `myid` file.

   d. In the other ZooKeeper servers, add the same server entries in the `zoo.cfg` file, create the `myid` file, and enter the corresponding server ID to the `myid` file.

Starting and Stopping a ZooKeeper Server

When you start the Identity Resolution servers, the ZooKeeper server starts . When you stop the Identity Resolution servers, the ZooKeeper server stops.

You can run the `zkServer` script located in the following directory to manually start or stop the ZooKeeper Server:

- On Windows: `<Identity Resolution Installation Directory>\bin`
- On UNIX. `<Identity Resolution Installation Directory>/bin`

Use the following command to manually start the ZooKeeper server:

```
zkServer start
```

Note: If you use an ensemble of ZooKeeper servers, manually start all the ZooKeeper servers before you start the Identity Resolution servers.

Use the following command to manually stop the ZooKeeper server:

```
zkServer stop
```

Environment Variables

The Console Server uses utility programs to perform tasks such as creating a system, loading an IDT, and running the batch search client. Some of these processes allow environment variables to control or alter their behavior. The utility programs inherit the server's environment variables.

Win32

Use the `<Identity Resolution Installation Directory>\env\isss.bat` file to set the server's environment variables.

UNIX

Use the `<Identity Resolution Installation Directory>/env/isss` script to set the server's environment variables.

Variable Descriptions

SSADB_QUERY_TIMEOUT

Sets the timeout interval in seconds for a search request to query the database.

SSADB_RECID_INCREMENT

Sets the increment value for the record identifiers. The increment value is applicable only for the synchronization process. Default is 1.

SSA_IGNORE_ODBC_SQLSTATE

Specifies the types of errors to ignore based on the SQLSTATE value. You can also specify a message code number to ignore the specific error.

For example: `SSA_IGNORE_ODBC_SQLSTATE=S1000,29725` ignores the following error:

```
SQLSTATE='S1000' NATIVE_ERR=29275 Reason: [Oracle][ODBC][Ora]ORA-29275: partial multibyte character
```

SSA_LITEINDEX_DONOTSEARCHNULLKEY

Skips the null values when you use Lite Indexes to search. To skip the null values, set the environment variable to any whole number. For example, `SSA_LITEINDEX_DONOTSEARCHNULLKEY=1`.

SSA_LISTEN_FAILURES_ABORT

Indicates whether to perform a forced restart of a server after the server exceeds the allowed number of consecutive connection failures.

Use one of the following values:

- `1`. Performs a forced restart.
- `0`. Disable this option. Default is 0.

SSA_LISTEN_FAILURES_ALLOWED

Maximum number of consecutive connection failures allowed for a server. Set the environment variable to any whole number. For example, `SSA_LISTEN_FAILURES_ALLOWED=10`.

SSANOSORTIDX

Indicates whether to disable sorting when you run the Table Loader utility. To disable sorting, set the environment variable to 1.

For example: `SSANOSORTIDX=1`.

SSAOPTS

Sets various logging and trace options. You can use one or more of the following values:

- `+r`. Logs all the search records to the `idssrsv.dbg` file. Use `+r` to identify a particular search transaction that causes a server crash.
- `+T`. Logs search trace information to the `idssrsv.dbg` file. You can use the LOGTEST trace file in addition to the search trace information in the event of a server crash.
- `+u`. Logs process resource usage, such as the number of threads, sockets, and stack space to the `*.dbg` files. The value also logs database resource utilization when users connect or disconnect.

For example, `SSAOPTS=+rTU`.

SSAPR

Directory name that contains the SSA-NAME3 population files.

SSA_RB_ERROR_IS_NOT_FATAL

Converts a data integrity error to a warning. To convert errors to warnings, set the environment variable to any whole number. For example, `SSA_RB_ERROR_IS_NOT_FATAL=1`.

SSA_RESTRICTED_VARS

Specifies a colon-separated list of environment variables which cannot be set by the console client.

SSA_SEARCH_MAX_RETRY

Maximum number of times you want to retry a search request before it fails.

SSASQLLDR

Fully qualified name of the loader utility, which is specific to each database. Use one of the following values:

- `sqlldr` for Oracle
- `db2` for IBM DB2 UDB
- `bcp` for Microsoft SQL Server

SSA_SOCKET_MAXIMUM_ALLOWED

Maximum number of sockets that listen for connections. Set the environment variable to any whole number. For example, `SSA_SOCKET_MAXIMUM_ALLOWED=10`. By default, you do not have any restriction on the number of sockets.

SSA_SOCKET_TIMEOUTS

Specifies the timeout periods as a comma-separated list for all the Identity Resolution servers. The SSA_SOCKET_TIMEOUTS environment variable uses the following format:

```
SSA_SOCKET_TIMEOUTS=<Idle>,<Connection>,<Write>,<Read>
```

Configure the following parameters:

Idle

Time period for a client session to remain idle before the server cancels the session. The default timeout period is 86400 seconds.

Connection

Time period for a client to wait before an attempt to establish a connection to the server is terminated. The default timeout period is 15 seconds.

Write

Time period to wait for a write or send operation to complete successfully. The default timeout period is 7200 seconds.

Read

Time period to wait for a read or receive operation to complete successfully. The default timeout period is 7200 seconds.

If you configure the SSA_SOCKET_TIMEOUTS variable, you must specify the timeout periods for all the operations. For example, `SSA_SOCKET_TIMEOUTS=86400,15,7200,7200`

SSA_THREAD_MAXIMUM_ALLOWED

Maximum number of threads that process the data. Set the environment variable to any whole number. For example, `SSA_THREAD_MAXIMUM_ALLOWED=10`.By default, you do not have any restriction on the number of threads.

SSATEMP

Some Identity Resolution programs and scripts require output to be written to a temporary directory. The location of this directory is controlled by the *SSATEMP* variable. The default location of this directory is `$HOME/tmp` in UNIX and `%TEMP%` in Windows installations. It is recommended that a separate location is created for each user (each instance or running servers). This directory must have write and execute permissions.

SSA_TREAT_C_TYPE_AS_LATIN1=1

Converts the Latin-1 character set to the UTF-8 character set.

SSA_USE_SQLDRIVERCONNECT

Indicates whether to use the `SQLDriverConnect` or `SQLConnect` function to connect to the target database. When you use the `SQLDriverConnect` function, you can specify multiple connection attributes.

To use the `SQLDriverConnection` function, set the environment variable to 1. By default, Identity Resolution uses the `SQLConnect` function.

The `SQLDriverConnect` function uses the following format for the connection string:

```
DSN=<Data source name>;UID=<User ID>;PWD=<Password>;<Parameter 1>=<Value 1>;<Parameter 2>=<Value 2>.
```

For example, the following sample connection string sets the minimum and maximum pool size:

```
DSN=<datasourcename>;UID=<userid>;PWD=<password>;Pooling=true;Pool Size Min=10;Pool Size Max=50.
```

SSA_USE_SQLDRIVERCONNECT_ATTRIBUTES

Specifies the additional attributes for the `SQLDriverConnect` function. Applicable only when you set `SSA_USE_SQLDRIVERCONNECT=1`.

For example, when `SSA_USE_SQLDRIVERCONNECT=1` and `SSA_USE_SQLDRIVERCONNECT_ATTRIBUTES=Pooling=true;Pool Size Min=10;Pool Size Max=50;Connection Lifetime=120;Connection Timeout=60;Incr Pool Size=5;Decr Pool Size=2`, Identity Resolution uses the following connection string:

```
DSN=<datasourcename>;UID=<userid>;PWD=<password>;Pooling=true;Pool Size Min=10;Pool Size Max=50;Connection Lifetime=120;Connection Timeout=60;Incr Pool Size=5;Decr Pool Size=2.
```

SSA_WSTIMEOUT

Specifies the timeout periods as a comma-separated list for the XML Search Server. The SSA_WSTIMEOUT environment variable uses the following format:

`SSA_WSTIMEOUT=<Idle>,<Connection>,<Write>,<Read>`

Configure the following parameters:

Idle

Time period for a client session to remain idle before the server cancels the session. The default timeout period is 86400 seconds. If you want to use the default timeout period, set the value as 0.

Connection

Time period for a client to wait before an attempt to establish a connection to the server is terminated. The default timeout period is 15 seconds. If you want to use the default timeout period, set the value as 0.

Write

Time period to wait for a write or send operation to complete successfully. The default timeout period is 7200 seconds. If you want to use the default timeout period, set the value as 0.

Read

Time period to wait for a read or receive operation to complete successfully. The default timeout period is 7200 seconds. If you want to use the default timeout period, set the value as 0.

If you configure the SSA_WSTIMEOUT variable, you must specify the timeout periods for all the operations. For example, `SSA_WSTIMEOUT=1200,0,0,0`

Note: If you configure both the SSA_SOCKET_TIMEOUTS and SSA_WSTIMEOUT environment variables, the SSA_WSTIMEOUT variable takes precedence over the SSA_SOCKET_TIMEOUTS variable for the XML Search Server.

SSA_ZOO_HOST=<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>]

Indicates a list of ZooKeeper servers based on the host name of the ZooKeeper server and the port on which the host listens for connections.

If you configure the ZooKeeper servers during the Identity Resolution installation, the SSA_ZOO_HOST environment variable is set with the ZooKeeper host details.

Note: Ensure that you specify 1, 3, or more ZooKeeper servers. You cannot achieve high availability with two ZooKeeper servers.

SSA_ZOO_PORT=<Port>

Indicates the port on which you want to start the ZooKeeper server.

If you specify the ZooKeeper port for the local host during the Identity Resolution installation, the SSA_ZOO_PORT environment variable is set with the ZooKeeper port.

ZOOCFGDIR=<Configuration File Directory>

Applicable only if you use an ensemble of ZooKeeper servers. Indicates the directory path to the configuration file of ZooKeeper named `zoo.cfg`. By default, you can find the `zoo.cfg` file in the following directory:

- On Windows: `<Identity Resolution Installation Directory>\bin`
- On UNIX. `<Identity Resolution Installation Directory>/bin`

ZOO_LOG_DIR=<Log File Directory>

Applicable only if you use an ensemble of ZooKeeper servers. Indicates the directory path to the log file for the ZooKeeper server.

Windows Services

You can create Windows services that run programs or batch scripts to start and stop Identity Resolution servers and other processes.

Use the following utilities to create a Windows service that runs a program or a batch script during a Windows service startup or shutdown process:

- `idssvc` for all the servers except the Synchronization Server.
- `updsvc` for the Synchronization Server.

Note: If you plan to start the Identity Resolution servers as Windows services, you must use a database client installed on a local drive instead of a shared network drive to connect to the database.

idssvc Utility

Use the idssvc utility to create or delete a Windows service for all the servers except the Synchronization Server. The idssvc utility uses the following syntax:

- To create a Windows service:

```
idssvc install <Service Name> <Start Program Name> <Stop Program Name>
```

- To delete a Windows service that you create:

```
idssvc delete <Service Name>
```

The idssvc utility uses the following parameters:

Service Name

Name of the service that you want to create. The idssvc utility adds the `IDS_` prefix to the service name to ensure that the Identity Resolution services group together when you view them in the Service Control Manager.

Start Program Name

Fully qualified name of a program or script to run when you start the service. Enclose the name of the program or script and its parameters within double quotes (").

Stop Program Name

Fully qualified name of a program or script to run when you stop the service. Include all the parameters that you specify in the `Start Program Name` parameter, and enclose them within double quotes (").

For example, the following command creates a Windows service, `licenseserver`. When you start the `licenseserver` service, the service starts the License Server, and when you stop the `licenseserver` service, the service stops the License Server:

```
idssvc install licenseserver "%SSABIN%\liup.bat" "%SSABIN%\lidown.bat"
```

Note: Informatica recommends that you use batch scripts instead of programs because a batch script can establish environment variables that Identity Resolution requires to function correctly. For example, when you set a service startup type to automatic, a batch script can set the environment variables before starting the Identity Resolution servers so that the servers function correctly.

After you create a Windows service, you can manually start or stop in the Service Control Manager or from a command line. To start or stop a service from a command line, run the following commands:

- To start a service, `net start <Service Name>`
- To stop a service, `net stop <Service Name>`

Note: When you specify a Windows service name, ensure that you include the `IDS_` prefix.

You can use the Service Control Manager to change the service startup type to automatic.

The <Service Name>.log file in the following directory logs all the error messages and informational messages associated with the service: <Identity Resolution Installation Directory\bin. The service does not log the messages to the Windows Event Log, which requires the provision and registration of a dependent message DLL file.

Sample Script to Create a Windows Service

You can find a sample script in the following directory: <Identity Resolution Installation Directory\bin\svcdemo.bat

The following example uses the sample script for both the start and stop calls. However, in the production environment, you can use different start and stop scripts. Specify the start parameter with the start script name so that the scripts do not run in the same shell.

```
idssvc install demo "c:\InformaticaIR\bin\svcdemo start" c:\InformaticaIR\bin\svcdemo
```

Use the following guidelines if you create a script that runs when you start or stop a Windows service:

- Call the following file to establish the Identity Resolution environment variables: <Identity Resolution Installation Directory>\env\isss.bat
- Start all the processes with the start parameter so that the processes do not run in the current shell. For example, if you start the Update Synchronizer without the start parameter, the service script cannot return the control until the Update Synchronizer stops.

  Note: The idsup.bat file located in the following directory uses the start parameter internally:

  <Identity Resolution Installation Directory>\bin
- Call other scripts, but ensure that you transfer the control by using a call parameter. Otherwise, the control never returns when the called script ends, and the service script cannot return the control.
- When you stop the servers by using the idsdown.bat file located in the following directory, you can specify the hard option to force an immediate shutdown that disconnects the active clients: <Identity Resolution Installation Directory>\bin

  For example, you can start the Update Synchronizer with the --rbcheck switch to periodically test the Rulebase connectivity and to abort the connection when the Rulebase Server is inaccessible. This switch avoids the need to run the syncstop script.
- Set the response code to indicate success or failure at the end of the script. Use the cmd /c exit /b %SSARC% command, which sets SSARC to 0 for success and 1 for failure.

updsvc Utility

Use the updsvc utility to create a Windows service for the Synchronization Server. Before you run the utility, you must update the values of the following environment variables in the <Identity Resolution Installation Directory>\bin\multistart.bat file:

- SSATOP=<Identity Resolution Installation Directory>
- SSA_SEHOST=<Search Server Host:Port>
- SSA_RBHOST=<Rulebase Server Host:Port>
- SSA_CSHOST=<Console Server Host:Port>
- SSAWORKDIR=<Absolute Path for the Log Files>

The updsvc utility writes the log messages to the sync.log and sync.err files when a service runs.

Use the following syntax to run the updsvc utility:

```
updsvc install <Service Name> "%SSABIN%\multistart.bat odb:0:userid/password@service <System>
<IDT>" "%SSABIN%\multistop.exe -p<System> -e<IDT> -h<Console Server Host:Port>"
```

Configure the following parameters:

Service Name

Name of the service that you want to create. The updsvc utility adds the `IDS_` prefix to the service name to ensure that the Identity Resolution services group together when you view them in the Service Control Manager.

odb:0:userid/password@service

Rulebase connection string. A rulebase connection string includes the rulebase number, the service name that Identity Resolution uses to refer to the database service, and the user credentials to access the database service. For example: `odb:99:ssa/ssa@ora920`

System

Name of the system to synchronize.

IDT

Name of the identity table that you want to process. Ensure that the identity table is available in the specified system.

Console Server Host

Host name of the Console Server.

Port

Port number on which the Console Server listens.

After you create a service, use the command line or the Service Control Manager to start the service.

To start a service from the command line, run the following command:

`sc start IDS_<Service Name>`, where `Service Name` is the name of the service that you want to start.

To stop a service from the command line, run the following command:

`sc stop IDS_<Service Name>`, where `Service Name` is the name of the service that you want to stop.

CHAPTER 3

Console Client

This chapter includes the following topics:

- Overview, 45
- Starting, 45
- Modes, 47
- Window Layout, 48
- Menu Items, 50
- System Editor, 57
- Log Viewer, 57

Overview

The IIR Console provides the user with centralized control of the various components that make up the IIR system.

The Console is a client/server application.

The Console server is a non-interactive program, which would normally run on the machine where the database resides. When it is run, the Console Server will establish its environment and then wait for clients to connect. Once one or more clients are connected, the server launches and monitors the progress of the various IIR programs at the request of these clients.

The Console client is a Java GUI program. It can be launched on any machine which is connected through TCP/IP to the Console Server's machine and which has a Java Runtime Environment.

Starting

This section provides information on how to start the Console client.

Starting from Shortcuts

Two (Windows) icons for the Console Client are placed in the SSA Program folder by the Installation process. Click the **Console Client**icon to start the client in read-only mode. This mode is used to run search clients while restricting access to System maintenance utilities.

Use the **Console Client (Admin Mode)** icon to allow update activity such as creating, deleting and loading Systems.

Starting from Command Line

Once the Console Server is running, the Console Client can be started using this command:

For Win32:

```
%SSABIN%\idsconc [-cWORKDIR] [-dX] [-hHOST] [-rhRBHOST]
[-rnRBNAME] [-pPROFILE] [-wWORKDIR] [-vVERBOSITY] [-a]
```

For Unix:

```
$SSABIN/idsconc [-cWORKDIR] [-dX] [-hHOST] [-rhRBHOST]
[-rnRBNAME] [-pPROFILE] [-wWORKDIR] [-vVERBOSITY] [-a]
```

where the optional switches are:

-a

starts the client in Admin mode which permits System maintenance. Omitting this switch will place the Console Client in a non-administrative mode.

-DX

X is the debug level and determines how much debug information will be logged. It must be in the range 0-3. 0 requests no debug information, while 3 requests that all debug information be logged. 0 is the default.

-cWORKDIR

This defines the name of the Work Directory to be used by client programs. This parameter is optional. If specified, it must specify a directory that is accessible to the machine on which the Client is running. At present, this parameter is used only by the Relate Client. If you are not planning to run the Relate Client, then there is no need to supply this parameter.

-hHOST

This parameter may be used to determine which Console Server the client will connect to. It should be in the form `host:port`, where `host` is the hostname or IP address of the machine where the Console Server is running and `port` is the port number on which the Console Server is listening.

The default value is `localhost:1669`.

-pPROFILE

This parameter may be used to define a Profile name. A profile is used to store session state information in the Rulebase. This allows a client to restart using the same settings as the previous time that profile was used. Using a profile can cause problems if you are planning to reinitialize the Rulebase or switch Rulebases mid-session. In such cases, use -p-to disable profiles.

-rhRBHOST

Optional parameter.

-rnRBNAME

These optional parameters may be used to set the initial Rulebase Host and Rulebase Name values for the client. If present these values will override any default values supplied by the Console Server.

-wWORKDIR

This defines the value for `SSAWORKDIR` to be used by the Console Server on behalf of this client. This is a directory on the machine where the Console Server is running.

-vVERBOSITY

This defines the default verbosity setting to be used.

Note: The case of option letters is significant.

Modes

This section provides information on the modes you can connect.

Configure Mode

Configure Mode is used to run the Installation tests.

When the Console Client is run, it interrogates the Server to determine the Server's mode of operation. If the Server is in **Configure** mode, the Client initiates the setup process by displaying several dialogs. The user is prompted to supply the required information such as the user's database name.

The user should fill in each of the required fields and then click **Finish**. The console will then go through the steps involved in completing the installation of IIR.

These include:

- creating and initializing a Rulebase
- running the standard tests

This serves to confirm that the IIR installation is working correctly. Upon successful completion, the Server and Client change to **normal** mode of operation. The Client can then be used to carry out normal IIR operations. There is no need to restart either the Client or the Server.

On the other hand, if an error should occur, the Server remains in **Configure** mode and the install process can be repeated if required.

Normal Mode

When the Console Client is started it connects to the Console Server determined by the -h parameter, if supplied, or to the default Console Server. It then determines from the Server the mode of operation.

If the mode of operation is not Configure Mode, the Client presents a dialog to the user that contains a list of user-settable variables. These variables are described below.

Rulebase Name

The name of the Rulebase to be used.

Work Directory

The name of the directory on the server's machine where output files will be placed. This field is mandatory. Note that this value can be set using the -w command line option.

Client Work Directory

This defines the name of the Work Directory to be used by Client programs. If specified, it must specify a directory which is accessible to the machine on which the Client is running. At present, this parameter is used only by the Relate Client. So, if you are not planning to run the Relate Client, then there is no need to supply this parameter.

Service Group Directory

When a new System is created, IIR will look in this directory for any required SSA-NAME3 v1.8 service groups (in the form of `.dat` files). This value can be overridden by a parameter in the **Create System** dialog. This parameter should be left blank if SSA-NAME3 v1.8 is not used.

Rulebase Server

The name of the host where the Rulebase Server to be used during this session is running.

Port

The port number on which the Rulebase Server is listening.

Connection Server

The name of the host where the Connection Server to be used during this session is running.

Port

The port number on which the Connection Server is listening.

Search Server

The name of the host where the Search Server to be used during this session is running.

Port

The port number on which the Search Server is listening.

Statistics

If selected, the log files will include statistics.

Usage Summary

Select this option to produce database usage statistics.

Server Trace

If selected the Console Server produces verbose output. This is for troubleshooting purposes and should normally be disabled.

Live Progress

Check this option to see the live progress every time an action is performed.

All the above variables are used by the Console Server to service requests from the Client. Therefore, care should be taken to see that the values are correct. The user should make any required changes and click **OK**. At this point, the Main Console Window is displayed.

Window Layout

The user may now make a selection from the various buttons to perform the desired task. The buttons are arranged in two groups. The row of buttons along the top of the Console window are associated with the various objects with which a user might want to work, such as System, Rulebase, etc. Click one of these buttons causes a second group of buttons to appear down the left-hand side of the Console window. These buttons are associated with various actions that can be carried out on the object selected from the first button group. Example, if the user click **Rulebase** then the possible actions will be **Edit** and **Create** and two buttons will appear in the second button group to allow the user to select the desired action. In addition, there is a group of four buttons at the bottom of the left hand panel. These buttons are independent of the top row of buttons and provide quick access to some basic functions.

In addition to the buttons there is a menu bar. In general, the options on the menu bar mirror those available through the buttons mentioned above.

To the right of the second button group is the messages panel. This is a read only area where Console will display progress and error messages.

Along the bottom of the window is the status bar. This contains the current settings for **Work Directory**, **Rulebase** and **System**.

Launched Jobs

This is a list of all the jobs launched during the current session. Each user can access more information about a particular job in the list. Click the **Open** button.

When reconnecting the client to the console server, the list will display all the currently running jobs for all console clients using the same Rulebase.

The progress messages for each job are not displayed automatically when a Client reconnects. The user must select a running job from the list and click **Open** (or double-click the item). This will open the usual progress window.

Options

Open

Opens a status window for the selected job.

Delete

Remove the selected job from the list. Note that only completed jobs may be removed from the list.

Refresh

Refreshes the list with the currently running jobs for the same Rulebase.

Server Status Indicators

Work directory

The name of the work directory on the server's machine where temporary and the output files will be placed.

Rulebase

Name of the Rulebase currently being used.

System Name

The name of the system in use.

Profile Name

The name of the profile in use.

Console Server Status

Indicates, if the Console Server is running or not.

✓ If Console Server is running.

X If Console Server is not running.

Search Server Status Indicates, if the Search Server is running or not.

✓ If Search Server is running.

X If Search Server is not running.

Connection Server Status Indicates, if the Connection Server is running or not.

✓ If Connection Server is running.

X If Connection Server is not running.

Common Toolbar Buttons

The following describes the functionality provided by the four buttons **Status**, **Settings**, **View Logs** and **Clear Messages**:

Server Status

This button activates the Status dialog, which reports the status of the IIR servers, the Rulebase and the database associated with the current system.

Settings

This option will display the dialog containing the current environment of the client. This is the same dialog as the one presented when the Client is first started. The user may make any required changes to the environment variables.

View Logs

Use this button to activate the Log Viewer. The Log Viewer allows the various output files produced by IIR to be viewed.

The Log Viewer displays the files in a Tree layout with the file size (rounded to the nearest kB) and indicates if a file is empty. The Log Viewer also gives the user the ability to delete individual logs as well as all the logs associated with the run itself.

Clear Messages

Click this button to clear the main message window.

Menu Items

This section describes about the menu items in IIR.

Servers Menu

Many of the following menu items refer to file names. IIR Console does not support spaces in file names; its behavior is undefined if such file names are used.

Menu item	Function
Start	Allows the user to start the IIR Servers.
Stop	Allows the user to stop the IIR Servers.
Status	Allows the user to determine the current status the IIR Servers.

Rulebase Menu

Menu item	Function
Select	Allows the user to switch between different Rulebases.
Edit	Invokes the Rulebase Editor to edit the current Rulebase, defined by Rulebase Name and Rulebase Server.
Create	Allows the user to create and initialize a new Rulebase.
Resync	Use this option to force the Console Client to resynchronize its connection to the Rulebase. This may be necessary if a batch script has been run which has altered the state of the Rulebase in any way. **Note:** It should not be necessary to use this option in the majority of cases. Only users who are running scripts which interact with IIR outside of the Console may need to do so.

Database Menu

Menu item	Function
Create	Allows the user to create and initialize a new Database.

System Menu

Menu item	Function	Parameters
New	Use this option to create a new System. A dialog will be presented allowing the user to indicate the source of the new System, which can be "SDF File" or "Clone the Current System". When a selection has been made, click **OK** and a appropriate dialog appears.	
Create System from an SDF	Allows the user to specify a `system.sdf` file. The Console Server then runs `sysload` to load the definitions in `system.sdf` into the Rulebase. This new System will then be added to the list of available Systems. The user must also supply the database name to be used during the System Load.	System Name: The name of the new system to be created must be specified here. This name must match with the name specified in the system definition file. (Mandatory parameter.) Definition File: Specify the name of the system definition file which describes the new system. Mandatory parameter. Database: The name of the database to be used by the system.

Menu item	Function	Parameters
Import a System from a flat file	Creates (restores) a System from a flat file which was created using the **System** > **Export**option. Systems can only be imported using the same software version there were exported with.	Input file: Specify the name of the flat file, which contains the System to be imported. Mandatory parameter. System name: Specify the name of the system to be imported, into the current Rulebase. This name may be different from the original System which was exported. Mandatory parameter. Match System name: Check this option to verify that the new system name, supplied by the user, matches the System name stored in the input file. Import As Template: Import normally restores all system rules including the status of all objects that have been implemented. This is analogous to a database "restore" operation. Specifying Import As Template instructs the process to remove information about implemented objects so that the System can be used as a template for a new System.
Clone the current System	Make a copy of the currently selected System. The new system is assigned a new, user-supplied, name and is given a status of "build".	New System name: Specify the name to be given to the new System. Mandatory parameter. Database: The name of the database to be used by the new System.
Select	The user can select a System from the current Rulebase. This System becomes the default System to be used in any subsequent operations, which require a System.	System Name: Select a system from the list of available systems in the current Rulebase, to be used as the default system.
Delete	The user can delete a System from the current Rulebase. Before an already loaded system can be deleted, its status must be changed from "Locked".	System Name: Select a system from the list of available systems in the current Rulebase, to be deleted from the Rulebase.
System Status	Displays the status of the current System and allows the user to change it. When performing an operation that is incompatible with an object's status (for example, refreshing a locked system) the Console will permit the user to automatically unlock the object for a single operation or, optionally, for the entire session. This makes it easy to prototype multiple system changes and load operations without the need to constantly unlock it.	Build: If the system status is set to "build" then it means that the system has not been loaded yet. Locked: Select this option, to lock the current system, and no changes can be done to it until it is unlocked. By default, the Table Loader will set the status to "Locked" after a successful load. Production: By selecting this option, the current system status will be set to "Production". No further changes can be made to the system. Test: By selecting this option, the current system status will be set to "Test". A test system can be modified. Prototype: This option sets the System status to "Prototype". No further changes can be made to the System including changing its status. Prototype Systems can only be copied to a new System (that is they can be used as a template). Users can not set Systems to this status.
Edit	This option allows the user to either edit a new system or continue editing of a previously edited system.	

Menu item	Function	Parameters
Export	Export an existing System's rules. This is usually done for backup purposes or to transfer the System rules to another Rulebase. The output is written in SDF format, which is useful when transferring clear-text rules to another Rulebase. Use **System** > **New** > **Create System** from SDF to load a system from an SDF file.	Output file: Specify the name of the file that will contain the exported system. Mandatory parameter.
Load	Load the system.	Load System: Check this option to load the selected system in the current Rulebase. Mandatory parameter. Load SSA-NAME3 SVG's: Check this option to load SSA-NAME3 v1.8 Service Groups (deprecated).
Export SVG	Export an SSA-NAME3 v1.8 Service Group from the current system to a flat file (deprecated).	Service Group: Select the name of the Service Group to be exported. Output file: Specify the name of the flat file, which will contain the exported Service Group.
Refresh	The user can delete all existing database objects created for this system (IDTs, IDXs and triggers). Before an already loaded system can be refreshed, its status must be changed from "locked".	

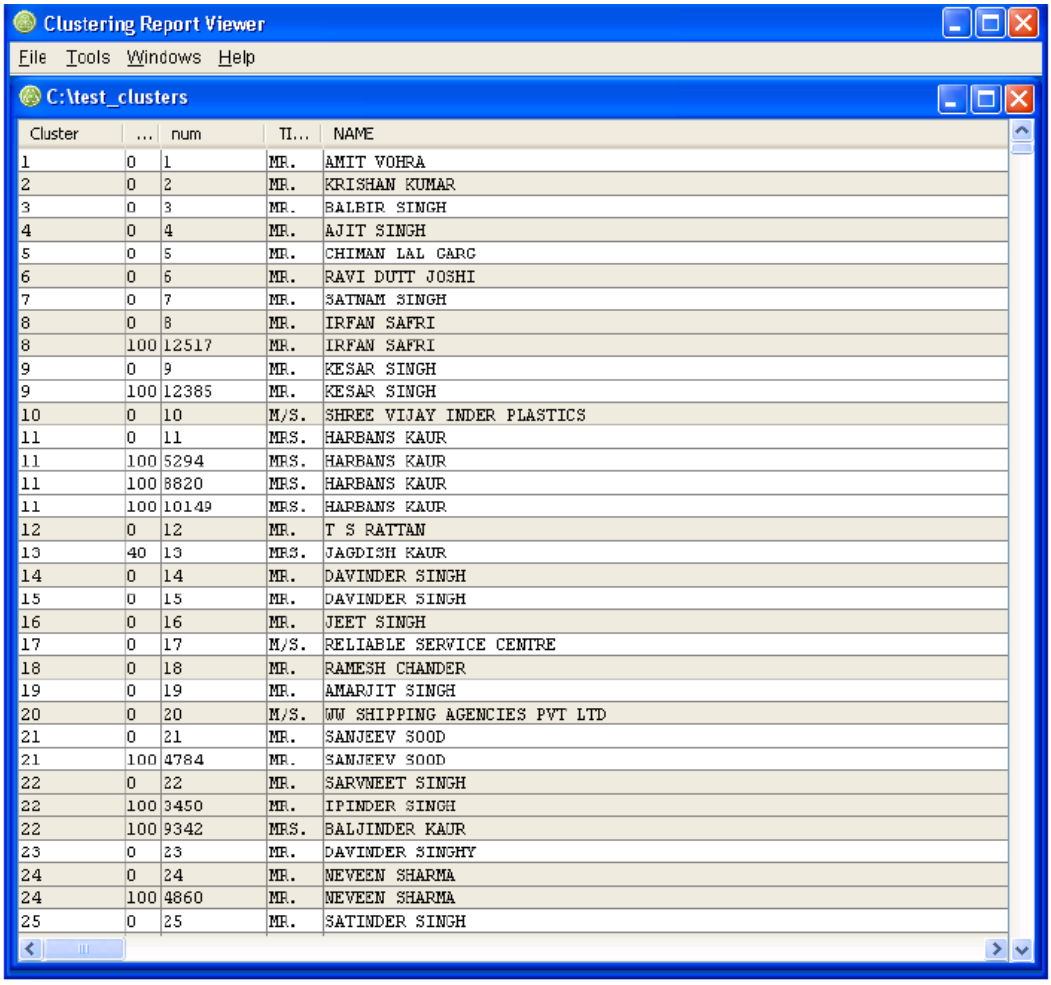
Tools Menu

Menu item	Function	Parameters
Search Client	Launches the IIR Search Client. The parameters passed to the Client will be the current System name, current Rulebase name and current Search Server address. Note that this option can be used to launch several Search Clients one after the other. So, by launching a Search Client, changing the default System name and then launching another Search Client it is possible to have two Search Clients running simultaneously but using different Systems.	
DupFinder	Run the `DupFinder` utility (that is relate in `DupFinder` mode).	*Output File* Specify the name of the file to store the matching records (all records not written to the optional `-m0` and `-m1` files). Mandatory parameter. *Search Definition* Select a Search Definition to be used. Mandatory parameter. *Search Width* Select a predefined search width. Narrow, Typical, Exhaustive or Extreme to be used. *Match Tolerance* Select a predefined match tolerance: Conservative, Typical or Loose to be used. *Output Format* Specify the report format. Values 0–7 are valid. *Starting Record ID* Specify the starting record id here. *Return Search records* Only Check this option to display only the duplicate search records. *Remove Search Record* Check this option to remove the search record from the resulting set. *Append New Line* Check this option, if a new line has to be append after each search record and viceversa. Valid only for output formats 0 and 3. *Trim Trailing Blanks* Check this option, if the blank spaces have to be trimmed and vice-versa. Valid only for output formats 0 and 3.
Run Clustering	Runs a selected clustering Once data has been loaded into an IDT, it may be clustered. This is the process of grouping like rows. For example, you may cluster by name in order to identify duplicates, or you may wish to cluster by name and address to identify "households".	Search Definition Select a Search Definition to use. Mandatory parameter. Singles Report File Specify the name of report file to have single-member clusters. Plurals Report File Specify the name of report file to have multi-member clusters. All Clusters Report File Specify the name of report file to contain single-member clusters (a.k.a. Singles), as well as multi-member clusters (a.k.a. Plurals).

Menu item	Function	Parameters
Load ID-Table	Allows the user to select a Loader-Definition. The Console Server then executes the utilities required to load the ID-Table.	*Loader Definition* Select the name of the loader-definition to be run.
Run Program	Runs a user-specified program on the server.	*Command Line* Specify the program to run followed by its parameters.
Relate	Runs Relate on the Server using the Search-Definition selected by the user.	
Execute SQL	Allows the user to specify a file containing SQL. The Console Server then invokes `ssaplus`, passing the supplied name as a parameter.	*Log On* Identifies the database against which the SQL should be run. *Default Logon* Check this option to use the default logon. *File Name* Specify the file containing the SQL to be executed.
Relperf	Run the Relative Performance Utility to compare search strategies.	Input File: Name of the file containing input records. Mandatory parameter. Output File: The name of the output file to use. Mandatory parameter. Search Definition: Select a Search Definition to be used. Mandatory parameter. Output View: Name of the output view to use. Input View: Nominates the view that describes the input records. If not specified then the IDT layout will be assumed. Text Report: Generated report with be in text format. The default is a tab delimited report ideal for use with spread sheets. Secondary Report: Generate a second report for each search ordered by Match Tolerances instead on SearchWidths. Alternate Report: Generate an alternative style report with a histogram of accepted count. Build Default Stats View: Check this option to generate a default statistical view for use during the `relperf` run. When using this option an output view does not need to be specified. If an output view is specified then the generated view will consist of all the fields in the specified view plus any fields in the default statistical view that are not already present.
Clustering Viewer	The Clustering Viewer can be used to view the results of various Clustering runs.	

Starting from the Console

Clustering Viewer can be started from the Console Client by selecting **Tools** > **Clustering Viewer**.



Cluster	...	num	TI...	NAME
1	0	1	MR.	AMIT VOHRA
2	0	2	MR.	KRISHAN KUMAR
3	0	3	MR.	BALBIR SINGH
4	0	4	MR.	AJIT SINGH
5	0	5	MR.	CHIMAN LAL GARG
6	0	6	MR.	RAVI DUTT JOSHI
7	0	7	MR.	SATNAM SINGH
8	0	8	MR.	IRFAN SAFRI
8	100	12517	MR.	IRFAN SAFRI
9	0	9	MR.	KESAR SINGH
9	100	12385	MR.	KESAR SINGH
10	0	10	M/S.	SHREE VIJAY INDER PLASTICS
11	0	11	MRS.	HARBANS KAUR
11	100	5294	MRS.	HARBANS KAUR
11	100	8820	MRS.	HARBANS KAUR
11	100	10149	MRS.	HARBANS KAUR
12	0	12	MR.	T S RATTAN
13	40	13	MRS.	JAGDISH KAUR
14	0	14	MR.	DAVINDER SINGH
15	0	15	MR.	DAVINDER SINGH
16	0	16	MR.	JEET SINGH
17	0	17	M/S.	RELIABLE SERVICE CENTRE
18	0	18	MR.	RAMESH CHANDER
19	0	19	MR.	AMARJIT SINGH
20	0	20	M/S.	WW SHIPPING AGENCIES PVT LTD
21	0	21	MR.	SANJEEV SOOD
21	100	4784	MR.	SANJEEV SOOD
22	0	22	MR.	SARVNEET SINGH
22	100	3450	MR.	IPINDER SINGH
22	100	9342	MRS.	BALJINDER KAUR
23	0	23	MR.	DAVINDER SINGHY
24	0	24	MR.	NEVEEN SHARMA
24	100	4860	MR.	NEVEEN SHARMA
25	0	25	MR.	SATINDER SINGH

The **File** menu will allow you to open a Post Report or a Database. You can open the same one several times if you wish. This makes it easier to visually compare different parts of the same report at the same time.

Jobs Menu

This section describes about the Jobs menu options.

Edit

This option allows the user to

- Define a new job
- Edit a predefined job and also
- Delete a pre-existing job

Run

This option allows the user to select and run a job, from a list of predefined jobs belonging to the system.

Parameters

- Job NameSelect a job name to run from the list of available jobs in the current system.

- Start FromStep Select the name of the step at which the job should start running. Steps previous to the one selected will be skipped.

System Editor

The System Editor is a GUI tool to create a new system and also edit an existing system. System editor has five options:

Load

Use this option to load the changes made to the system.

Add

Use this option to add a new definition to the system.

Clone

Use this option to clone an existing definition in the system.

Delete

Use this option to delete an existing definition in the system.

Close

Use this option to close the editor.

Refer to the *Editing a System* section of the *DESIGN GUIDE* for more details on using the System Editor.

Log Viewer

Every time a procedure such as `Load-IDT`, `Relate` or a user-defined Job is started, a Run Number is assigned to that run and all relevant information is stored in the Rulebase. This information includes the Completion Status and details of any output files created during the run. The Run Number is used to uniquely identify the run.

The Log Viewer provides the user with the ability to access the run information for previously run jobs. There are two classes of Jobs; `System Jobs` and `Global Jobs`. `System Jobs` are jobs that are run against a particular system, such as Relate. `Global Jobs` are jobs that are not run against a particular System. These jobs either involve more than 1 System (example, Clone System) or are responsible for setting up a System (example, Create System).

Choosing The Run To Be Viewed

Select the type of Job, either `System Jobs`or `Global Jobs`, using the Radio buttons. If System Jobs is selected, select the required System using the dropdown list of Systems. If Global Jobs was selected, then the System need not be selected. Now choose the job name from the dropdown list of jobs. User-defined jobs are identified by their user-assigned names. Other procedures, such as `Relate`, are identified by the procedure name surrounded by asterisks. Example, *Import System*. An exception to this rule is `Load IDT` for which the name of the Loader-definition will be used, again surrounded by asterisks. Example, *table-1*.

When the job has been selected a list of runs for this job will be listed on the left-hand side of the Log Viewer. The runs will be sorted in ascending order, so the most recent run will appear at the bottom of the list. The

title for each run consists of the date and time when the job started and the Run Number which was assigned to this run. Select the run in which you are interested.

A list of the output files created by this run will appear below the list of runs. The most recently created output file will be automatically displayed in the right hand pane of the Log Viewer. To view other files in the list simply click the required file in the tree display.

Note: The Log Viewer will truncate (for display only) log files larger than 960KB.

Other Functions Provided By The Log Viewer

Delete File

Use this option to physically delete the currently selected file.

Delete Run

Use this option to delete all output files and run information for the currently selected run.

Refresh

Use this option to reread the run information from the Console Server. This option is useful if a job is currently running and you want to check if anymore output has been created.

Close

Use this option to Close the Log Viewer and return to the Main Console Window.

CHAPTER 4

Search Clients

This chapter includes the following topics:

- Overview, 59
- Deployable Search Clients, 60
- Administrator Search Clients, 61
- HTTP Search Client, 64
- Relate, 66
- DupFinder, 74

Overview

IIR provides several "out of the box" search clients that may be used as soon as a system has been defined and loaded. Although you are free to create your own customized search clients using the IIR Search API, these clients provide facilities to quickly utilize and/or deploy search functionality without any coding.

Online Search Clients

Online Search Clients dynamically adjust their dialogues based on the Search-Definitions defined in the Rulebase. There are three main categories of online clients:

Deployable search clients are restricted in functionality. They are used to quickly deploy fixed search capabilities to end-users. These clients cannot switch between searches or change search strategies.

Administrator search clients are intended to be used by an IIR Designer / Administrator. They are functionally rich and provide access to all searches and search/match strategies defined in a system. They also contain tracing and debugging facilities to help tune searches.

Web: Any web browser that is pointed to the IIR HTTP Server can act as a search client.

Batch Search Clients

IIR provides two "out of the box" batch search clients:

Relate runs a number of searches using search records selected from an input file or database table. Results are written to an output file.

`DupFinder` identifies duplicate records in the IDT and writes results to an output file.

Deployable Search Clients

Deployable search clients are designed to help IIR Administrators to quickly deploy search functionality to end-users. These clients are specifically designed to only run a pre-defined Search so that end-users cannot change search and matching strategies.

Java Applet

A Java Applet suitable for embedding within an HTML page is available as a Web based client. A Java enabled browser must be used to run the applet. A Java plug-in of version 1.4 or higher is recommended.

Parameters

The following parameters are mandatory:

HostPort

The Port number that the IIR Connection Server is listening on. Note that the Connection Server and Web Server must run on the same computer.

RulebaseName

The name of the Rulebase

WorkDirectory

The working directory.

System

The name of the System to open at startup.

Search

The name of the Search to open at startup.

HTML

The following HTML code snippet demonstrates how to instruct a browser to load the applet. The applet's code resides in a JAR file that is in the same directory as the HTML document. This example sets the initial size of the applet to 800 x 600 pixels.

```
<APPLET ARCHIVE="sclient.jar" CODE="ssa.clients.sclient.SsaClient"
WIDTH=800 HEIGHT=600>
<PARAM NAME="HostPort" VALUE="1667">
<PARAM NAME="RulebaseName" VALUE="odb:0:ssa/ssa@ora817">
<PARAM NAME="WorkDirectory" VALUE="c:\InformaticaIR\ids">
```

An example HTML is provided in `samples/programs/applet/SimpleSearchClient.html`.

Usage

To initiate a search, enter the required data and click the **Search** button.

The output results are shown in a table format and may be customized to reorder, resize and/or hide columns. Columns are reordered and resized by dragging their titles. Columns can be hidden or reenabled by right-clicking on the output table and selecting the appropriate option on the pop-up menu.

You can double-click on a specific record in the output table to perform a new search.

Administrator Search Clients

Administrator search clients are for the exclusive use of Identity Resolution System Administrators. They provide a rich set of features used to test and tune search strategies.

Identity Resolution provides two administrator search clients:

- A default search client that contains facilities to expand records and save history and to start a new search using the previous search results. However it does not work with multi-byte character sets or UNICODE data.
- A Lite client that supports multi-byte and UNICODE data. It also contains facilities to trace client side search data. This is particularly useful when debugging searches containing multi-byte data.

Starting from the Console

You can start the Administrator Search Client from the Identity Resolution Console Client by selecting **Tools** > **Search Client**.

Starting from a Shortcut

On windows, select **Search Client** from the **Informatica** program group.

Starting from the Command Line

To start the Administrator Search Client from the command line, run the following command:

- On Windows:

```
%SSABIN%\ssacs [-h<Host>:<Port>] [-s<Search>] [-w<WorkDir>] [-p<System>]
 [-r<Rulebase>]
```

- ON UNIX:

```
$SSABIN/ssacs [-h<Host>:<Port>] [-s<Search>] [-w<WorkDir>] [-p<System>] [-r<Rulebase>]
```

Configure the following parameters:

-h<Host>:<Port>

Name of the host and the port number of the Connection Server.

-s<Search>

Name of the search.

-w<WorkDir>

Absolute path for the working directory.

-p<System>

Name of the system.

-r<Rulebase>

Rulebase connection string.

The scripts also use environment variables to set some parameters. You can alternatively set the following parameters through environment variables:

RulebaseName

Name of the Rulebase. Specify the SSA_RBNAME environment variable.

ConnectionServer

Name of the host and port number for the Connection Server. Specify the SSA_COHOST environment variable.

WorkDirectory

Specify the SSAWORKDIR environment variable.

Client INI file

Some parameters for the client can be set using an INI file called `idsclie.ini`.

Note: The following parameters are specified on the command line in the client starting script ssasc: ADDRESS, PORT, SEARCH, SYSTEM , RULEDB, WORKDIR. Unless these parameters are not removed from the script the corresponding settings in the INI file are ignored.

The following parameters are common to both clients:
RULEDB= the Rulebase (default 0:system/manager).
ADDRESS= the Host name of the IIR Connection Server.
PORT= the Host port of the IIR Connection Server.
SEARCH= the initial Search to open (optional)
SYSTEM= the name of the System to open (default is "test").

The following parameters are for the default Search Client:
SIZE= the initial size of the window (default 1100x820)
FONT= the initial font of the tables (default Courier).
FONTSIZE= the initial font size of the table (default 12).
DIV_LOCATION= the location of the divider between the history pane and the current pane of a search (default 0.5).
REFRESH_RATE= the number of records which will be loaded before the screen is refreshed (default is 2).
OUI_VERBOSITY= the verbosity of an `OpenUser` call.

If an `ADDRESS` is not specified the IIR Client will display a dialog box to obtain the Host Name of the IIR Search Server. Similarly if a Search is not specified the Client will start without opening any IIR-ID Tables.

Note: If any of the text which follows the = character in each line of the INI file contains certain characters, then those characters must be preceded by a \ (backslash) character. These characters are:

- #(hash)
- !(exclamation)
- =(equals)
- :(colon)
- \(backslash)

For example, to specify a Rulebase whose name is `odb:0:SSA12/SSA12@SSA19`

the `idsclie.ini` file should contain this line:

```
RULEDB=odb\:0\:SSA12/SSA12@SSA19
```

Client Selection INI file

When the Administrator Search Client first starts, a dialog is presented that allows the user to select between these two clients. The choice is stored in the client selection INI file's (`adminsc.ini`) SMODE parameter. The mode may be switched using **Options** > **Search Client Selection**menu on the `Default` client and through the **Client Selection** button on the `Lite` client.

`SMODE=` the client mode (1=`Lite` search client, 2=`Default` search client)

Default Client

Upon startup, the Client prompts for the ADDRESS of the Server. Enter the Host name (or IP address) of the IIR Search Server. If left blank, the Host name defaults to the name of the machine running the Client.

If the IIR Client is started without an initial Search you must choose a Search either from the Search menu or from the Searches on the search toolbar. With a Search selected the program will open a new window for the IIR-ID Table. Each window represents one IIR-ID Table, thus if multiple searches are defined on the one IIR-ID Table it will not open a new window. However, if a Search is selected which is not defined on any of the currently open IIR-ID Tables a new window will be opened.

With a window open and a Search selected you will be prompted to input the required parameters for the Search. You can press**Enter** or click the **Search** button to perform a search. The data will be loaded and sorted by score. Select the **STOP** button, it will cause all current searches being loaded to stop.

If you wish to input values from an existing search simply select the field either before selecting the **Search** button (or menu) or whilst the search panel is displayed.

To zoom view a record, double-click the**record**. Right-click the **record** to bring up a pop-up menu with various options of that record.

Note: The **Print** menu option doesnot currently work, due to limitations in Java.

You can dynamically resize or reposition any of the columns in the table view simply by either "grabbing" a column header and repositioning it or dragging the border of the header in order to resize it. Furthermore there is a menu option under **Layout** > **Define Layout**which enables you to configure which columns are visible.

Under **File** there is the option to **Save** an output file. There is also the option to **Zoom All records** or **Dismiss All zooms**.

There is a scrapbook that enables copying any relevant records for later perusal. You can launch searches from these records in a similar fashion as from the main window.

This client's INI file is created/updated whenever this client shuts down.

Lite Client

At startup, the client presents the user with a list of available Systems. The default selection will be the first available System in the Rulebase.

If you want to swtich between Systems and Searches by, click the **Options** button.

The **Options** dialog also allows the user to fine-tune the Search Widths and Match Tolerances. The options presented may vary, as they are dependent on the System and Search definitions.

After you enter the required data, to initiate a search, click the **Search** button.

The output results are shown in a table format and may be customized to reorder, resize and/or hide columns. Columns are re-ordered and re-sized by dragging their titles. Columns can be hidden or re-enabled. To do this, right-click the output table and select the appropriate option on the pop-up menu.

A new search can be performed using a specific record in the output table. To do this, double-click that record.

HTTP Search Client

IIR supports the use of an Internet Browser as a search client. Web pages containing dynamically generated search dialogues based on your Systems are served up by the IIR HTTP Search Server.

Simply point your browser at the HTTP Search Server by typing its `host:port` in the Location Bar and follow the prompts. The default port number of the HTTP Search Server is 1672.

If prompted for a Rulebase name, the client must supply the information using a Dictionary Alias without the `ids: prefix`. See the *Dictionary Alias* section for more information. This is the only acceptable form of Rulebase name and is necessary to avoid passing clear text passwords to the server. To avoid the need for Rulebase names altogether, the administrator should define them in the HTTP Search Server's `.ini` file. Refer to the Configuring section below.

You do not need to enable any active content facilities such as Javascript. The Web pages are compatible with Netscape 4, Internet Explorer 5 and Firefox 1.0 (or later versions).

Configuring

The HTTP Search Server will not start unless it has been enabled and configured.

It is enabled by allocating the server's host name `SSA_HTHOST` and port number `SSA_HTPORT` in the `env\isss.bat` (Windows) or `env/isss` (UNIX) scripts.

The configuration process consists of creating a simple text file named `htserv.ini`. The file can be located in `$SSAINI`, `$HOME` or `$SSABIN`, which are searched in that order.

The content of this file determines which Searches and Rulebases are visible to the Web client. It is read at server initialization, so changes to the configuration become effective only after the HTTP Search Server is bounced. Lines starting with a semi-colon (;) are treated as comments. White space in the section headings (example, `[profile:basic]`) is not permitted, except as part of a name.

Generic Mode

The simplest possible file contains the following lines:

```
[Server]
mode = generic
```

This directs the HTTP Search Server to prompt the client for a Rulebase name and does not restrict access to any systems or searches. The optional line

```
rulebase = <dbtype>:<dbid>:<uid>/<pwd>@<svc>
```

may follow the `mode` line to specify the Rulebase to use. When provided, the client will not be prompted for this information. When omitted, the HTTP Search Server will request the client to enter the name of the Rulebase.

Note: Rulebase names are sent from the client to the server in clear text using the HTTP protocol. To avoid passing database passwords, clients must specify `Dictionary Alias` names without the `ids: prefix`. The HTTP Search Server assumes all Rulebase names received through HTTP are Dictionary Aliases and automatically prefixes them with `ids:` before use.

Custom Mode

Custom mode is use to configure the Systems, Searches and Rulebases visible to the Web client. When the HTTP Search Server runs in custom mode.

```
[Server]
mode = custom
profiles = basic
```

the Web client will offer the choice of one of a number of predefined profiles, that have been defined to allow access to a specific Rulebase, System and Search(es).

For example,

```
[profile:basic]
rules = just_one_search
[rule:just_one_search]
rulebase = odb:0:ssa/ssa@ora10g
system = testx216
searches = Claimant Names
[search:Claimant Names]
sdf-search = claimant-names
sdf-view = names_idt216
```

The example defines one profile named `basic`, however, multiple profiles can be specified by listing them as a comma separated list. Each profile may contain one or more rules, listed with the rules parameter (which is a comma separated list). In this case, there is just one rule named `just_one_search`. Each rule must have a corresponding definition that nominates the Rulebase name, System name and a comma separated list of Searches (`Claimant Names` in this example) that can be used by a user of this profile.

Note: The search names may contain spaces. This is allowed for aesthetic reasons, as the Search names are displayed by the Web client.

Each `search` must have a corresponding definition that nominates the name of the Search-Definition in the SDF (`claimant-names`). It may also optionally nominate an output view.

Output views are useful in that they can be used to define the order and/or columns displayed by the Web client. They can also add extra statistical information such as the Score or the number of the Multi-Search that returned the data.

A more complicated configuration may provide a second profile for advanced users too. For example:

```
[profile:advanced]
rules = just_one_search,other_searches
[rule:other_searches]
rulebase = ids:rb
system = testx217
searches = All Names
[search:All Names]
sdf-search = all-names
```

Profiles, rules and searches may be defined in any order, and must be defined if referenced.

Operation

Upon connection, the HTTP Search Server will prompt you for the Rulebase, followed by the system and search you would like to perform. When selecting the rulebase, note that the user must specify a `Dictionary Alias` name without the `ids:prefix`. This avoids having to give users access to the database password. This requirement is not available in `Custom Mode`, as the Rulebase must be specified in the config file.

Once in the search page, users can switch between different searches available to them. To switch between tabs, click the tabs at the top of the page.

To perform a search, the user must enter the appropriate information in the search fields on the top left of the page, then press **enter**, or click the **Search** button. The search data and browser encoding should be in UTF-8 format.

If the results are too wide, or too long to fit within the browser view port, scrollbars will appear indicating that more data is available. These scrollbars will scroll the results only - not the whole page. You can centre the results in the viewport so that they fill the whole browser. To centre the results, click the result summary (`56 results found for ...`).

You can manually override the prompt used by setting an environment variable with the desired label. This must be defined in the execution environment of the Search server. For example,

```
SSA_CNDS_LABEL=MySearch
```

Relate

`relate` is a batch search application that accesses the IIR Search Server using the standard Search APIs.

It reads an input file containing search transactions. Each search transaction is passed to the Search Server which uses the nominated Search Definition to find matching records. These are written to the output file by relate.

Input records must be separated by a newline. By default their format must match the layout of the IDT to be searched. If the format differs from the IDT layout, the -iswitch can be used to nominate an input format. Multiple output formats are supported. These are controlled with the -o switch.

Normally all records returned by the Search Server are written to the output file. That is those records that have an acceptable score as determined by the Search Definition. Additional filtering is possible with the -l and -u switches. These are used to set upper and lower bounds for acceptable scores and are applied by relate prior to emitting records to the output report. Note that filtering with -l and -u is not integrated with the other options. The filter will remove all records that are not within bounds, even if this will result in an empty set, irrespective of whether or not non-empty sets were requested. For example with the -x or -m switches. These filters are primarily used to experiment with score thresholds. Once correct thresholds have been determined, add them to the Search Definition and discontinue the use of -l and -u.

The -m switch can be used to create multiple output files.

`relate` can also search for duplicate records in the IDT. When started with the -x switch, relate runs in DupFinder mode.

Note: The input and output files need to be local to where the `relate` process runs. If relate is started from the command line, the files must be addressable from the same machine. If relate is started through the **Console Client**, it will run on the same machine as the Console Server.

Starting from the Console

Relate can be started from the IIR Console Client by selecting **Tools** > **Relate**. The utility is started by the Console Server and therefore runs on the same machine as the Console Server. The input and output file names must use paths that are valid within the context of that machine.

Starting from the Command Line

You can start Relate from the Command line on Windows or UNIX.

Use the following syntax to run Relate on Windows:

```
%SSABIN%\relate Search Infile Outfile -rRulebase -pSystem -hHost:Port -wWorkDir [Optional
Switches]
```

Use the following syntax to run Relate on UNIX:

```
$SSABIN/relate Search Infile Outfile -rRulebase
-pSystem -hHost:Port
-wWorkDir [Optional Switches]
```

Use the following options when you run Relate:

Search

Nominates the Search Definition to use.

Infile

The name of the file containing input records. When reading records from an SQL database, specify `lfile=xxx` where `xxx` is the name of the Logical-File-Definition that describes the SQL source. The same applies when reading records from an XML file. Specify `lfile=xxx` where `xxx` is the name of the Logical-File-Definition that describes the XML file.

Outfile

The name of the file that contains the matching records. These records are not written to the optional -m0 and -m1 files.

--append-to-output-file

Optional. Indicates to append the matching records to the output file. By default, the Relate process overwrites the output file.

-rRulebase

Required. The name of the Rulebase.

-pSystem

Required. The name of the System.

-hHost:Port

Required. The name of the host and the port number of the Search or Connection server.

-wWorkDir

Required. Work Directory.

-m0File

Optional. The name of the file to hold records that had no match.

-m1File

Optional. The name of the file to hold records that had one match.

-b

Optional. A binary input file containing records of fixed length. The record length must match IDT record length or Input View length.

-cOutputViewName

Optional. Nominates the name of the output view used to format the records returned by the search.

-dd<c>

Optional. Field delimiter character.

-df<c>

Optional. Field separator character.

-dr<c>

Optional. Record separator character.

-dl

Optional. Record layout.

-eEncoding

Optional. Nominates the UNICODE encoding used for W fields. The valid values are:

- 6 = UTF-16 Little Endian
- 7 = UTF-16 Big Endian
- 8 = UTF-8

--failed-searches-log

Optional. Absolute path and file name for the log file to log details about the failed searches.

--failed-searches-count

Optional. Maximum number of allowed searches that can fail. Default is 0.

--skip-input-records

Optional. Number of input records that you want to skip. Default is 0.

--query_timeout

Optional. Set the timeout interval in seconds for a search request to query the database.

-Ffilter

Optional. Nominates a single dynamic SQL Filter. For individual searches within a Multi-Search, this switch does not support multiple filter values.

-iInputViewName

Optional. Nominates the view that describes the input records. The default is the IDT layout.

-jSearchWidth

Optional. Nominates a predefined search width that overrides the width in the Search Controls: `Narrow`, `Typical`, `Exhaustive` or `Extreme`. You cannot use this parameter with a Multi-Search.

-kMatchTolerance

Optional. Nominates a predefined match tolerance that overrides the tolerance specified in the Search Controls: `Conservative`, `Typical` or `Loose`. You cannot use this parameter with a Multi-Search.

-nx[:y[:z]]

Optional. Use `x` search threads with an input queue of `y` records and an output queue of `z` records for each thread.

--retry-options=<Number of Retries>,<Time Interval>,<Number of Retries for Intermittent Errors>

Optional. Sets the following values:

- Maximum number of times you want to retry a search request before it fails. Default is 5.
- Time interval in seconds for the first retry attempt. The time interval doubles with every retry attempt. Default is 5.

- Maximum number of retries for any intermittent errors, such as socket error. Default is 5.

-s

Optional. Create a histogram of search transaction durations.

-ss

Optional. Provide individual timings for each search transaction.

-t

Optional. Append `newline`. The supported output formats are 0 and 3.

-tt

Optional. Append `newline` and trim trailing blanks. The supported output formats are 0 and 3.

-Tnum[,score]

Optional. Limit to num the number of records written from a result-set to the output file. Optionally, write more records than num if the records have a Score that is equal or greater than score.

-on

Optional. Specifies the report output format.

-ln

Optional. n = Lower score limit. Default is 0.

-un

Optional. n = Upper score limit. Default is 100.

-v

Optional. Verbosity level.

-Vpackage:parm

Optional. The VPD context-setting package and corresponding parameter.

-y

Optional. Print output view information at the start of the output file. Requires you to specify -c.

-x[{n|s}][rpt,recid]

Optional. `DupFinder` mode.

-X

XML output.

Report Formats

When you write to output files, choose a report value based on the output format that you want.

The following table describes the report format for each report value.

Report Value	Report Format

0	Prints each file record found on a new line. The following text shows sample output based on the report value:
	`JobN000005A ALGER COLLINS`
	`JobN000032A COLLINS`

1	Prints a line for each file record returned that consists of the search record, a nine digit search number that corresponds to the number of the search record in the input file, a three digit score, and the file record. Report prints nothing if no match is found. The following text shows sample output based on the report value: `JobN000005A ALGER COLLINS#000033064 100 JobN000005A ALGER COLLINS JobN000005A ALGER COLLINS#000033064 099 JobN000032A COLLINS`
2	Prints a group of records. The first line consists of the search record and then the search number. This is followed by one line per file record. Each file record is indented and is followed by the search number and score. Report prints nothing if no match is found. The following text shows sample output based on the report value: `JobN000005A ALGER COLLINS #000033064 JobN000005A ALGER COLLINS #000033064 100 JobN000032A COLLINS #000033064 099`
3	Prints the search record and a set of file records. A search number, surrounded by asterisks, precedes the search record. A three digit score precedes each file record. Report prints nothing if no match is found. The following text shows sample output based on the report value: `********** 000033064 ********** ---JobN000005A ALGER COLLINS} 100JobN000005A ALGER COLLINS} 099JobN000032A COLLINS`

4	Prints the search number and the file record on the same line. Report prints nothing if no match is found. The following text shows sample output based on the report value:
	`00033064JobN000005A ALGER COLLINS`
	`00033064JobN000032A COLLINS`

5	Prints the best file record for each search. You cannot specify a single match file (-m1). If you do not specify -m0, the output file contains the search record followed by the best file record, both on the same line. Report prints nothing if no match is found. If you specify -m0, the output file contains the best record, if any match is found. Else, the record is written to the unmatched file. The following text shows sample output based on the report value: `JobN000005A ALGER COLLINS JobN000005A ALGER COLLINS`

Report Value	Report Format

6	Prints the search number and the file record on the same line. Also sets the search number to zero. The following text shows sample output based on the report value:
	`00000000JobN000005A ALGER COLLINS`
	`00000000JobN000032A COLLINS`

7	Prints the best file record and the score returned for each search. You cannot specify a single match file (-m1). If you do not specify -m0, the output file contains the search record followed by the best file record, both on the same line. Report prints nothing if no match is found. If you specify -m0, the output file contains the best record, if any match is found. Else, the record is written to the unmatched file. The following text shows sample output based on the report value: `100JobN000005A ALGER COLLINS JobN000005A ALGER COLLINS`

Threads

Relate can run in a multi-threaded mode when the -n option is specified. Each search thread will independently connect to the Search Server and process searches in parallel.

There are two additional parameters associated with the -n switch: input queue and output queue.

The input queue specifies the length of queue that each thread will use to store the search records in. This queue must be long enough to allow the thread not to wait for I/O on the local relate input file. In general the default of 100 will be ample.

The output queue specifies the length of the queue that will hold each search thread's results. If any individual searches are expected to generate many matches, increasing the output queue size may improve performance.

Note: The output order of duplicate sets in a multi-threaded DupFinder report is dependent on the number of threads used to create the report.

SQL Input

`relate` can read input records from an SQL database instead of a file. In order to do this you must

- define source table(s) in the UST Section of the SDF using the `define_source` clause
- create a Logical-File Definition with `INPUT-FORMAT=SQL`
- run `relate` with the input file parameter set to `lfile=xxx` where `xxx` is the name of the Logical-File Definition.

The source definition should match the layout of the IDT (same field names, offsets and lengths). If it does not, use the `-i`switch to specify an input view so that the Search Server will convert the input record into IDT format prior to searching.

Note: A `define_source` clause automatically creates an input-view with the same name as the source.

XML Input

`relate` can read input records from a XML file instead of a flat file or SQL database. In order to do this you must:

- define source table(s) in the UST Section of the SDF using the `define_source` clause

- create a Logical-File Definition with `INPUT-FORMAT=XML`
- run `relate` with the input file parameter set to `lfile=xxx` where `xxx` is the name of the Logical-File Definition.

The source definition must match the layout of the IDT (same field names, offsets and lengths - and XML tags are case sensitive). If it does not, you can specify an XSLT clause, which is a reference to another XML logical-file-definition, which must be a valid XSLT stylesheet. This can be used to transform the XML input file into the required form.

For example:

```
logical-file-definition
*======================
NAME=                              lf-relate-xml
INPUT-FORMAT=                      XML
PHYSICAL-FILE=                   "+/data/relate.xml"
XSLT=                              lf-input-stylesheet
*
logical-file-definition
*======================
NAME=                              lf-input-stylesheet
COMMENT=                         "input stylesheet for initial load"
PHYSICAL-FILE=                   "+/data/relate.xsl"
FORMAT=                            XML
*
```

Delimited Input

Relate can read delimited input files. The field delimiter, field separator and record separator are defined with the -dd, -df and -dr switches respectively. They may specify a printable character or an escape sequence such as `\n` or `\x0a`. The default values are:

- Field delimiter `-dd"`
- Field separator `-df,`
- Record Separator `-dr\n`

Note: When using a UNIX based operating system, it is best to use hexadecimal values to define the delimiters, as certain ASCII characters have a reserved meaning and must be "escaped" by preceding them with a backslash (\) character.

The delimited data must be transformed into a format that matches the input view used by relate (specified with the -iswitch). If no input view is used, the delimited data must be transformed into IDT layout. Having determined the input view that will be used, the -dlswitch is used to describe how to transform the delimited data into that format. It specifies a comma-separated list of triplets:

```
-dl<triplet>,...
```

where each `<triplet>` consists of

- Field length (in printable decimal digits),
- R/L justification (optional, if omitted L is the default),
- Filler character preceded by a dash (optional, if omitted the default is a blank). It may be specified using an escape sequence.

The following example defines three fields. The first is 30 bytes long and uses the default justification and filler. The second field is 10 bytes, right justified and filled with 0. The third field is 50 bytes in length, left justified and filled with '!' (ASCII 0x21).

```
-dl30,10R-0,50L-\x21
```

The triplets are used by the transformation engine to convert the delimited data. The field lengths must match the length and order of fields in the input view. If a delimited field is longer than the field length, it will be truncated. If it is shorter than the field length, it will be either left or right justified and padded with the filler character up to the specified field length.

DupFinder Mode

Relate performs the `DupFinder` function when you start it with the -x switch.

For more information about DupFinder, see *Informatica Identity Resolution Developer Guide*.

The `DupFinder` function uses the following syntax:

```
-x[s|n][rpt,recid]
```

The `DupFinder` function uses the following options:

s

Returns search record in set. Required if you do not use the `n` option.

n

Does not return search record in set. Required if you do not use the `s` option.

rpt

Maximum number of times to call **ids_search_dedupe_start** (0=unlimited). Use it with the DEDUP-PROGRESS= parameter in the Multi-Search Definition to return after processing `DEDUP-PROGRESS` records.

recid

Starting record identifier for the search process. The default identifier is 0, which indicates the beginning of the IDT.

Output View Layout

IDT records can be returned using an output view by specifying the -cswitch. Adding the -y switch will print the output view layout in the beginning of the report file.

The format of this output view layout is as follows: The first line indicates the name of the output view. The second line gives title information. Each line after this gives details of a single field in the output view. These details are the ordinal number, name, offset, length and format of each field. For example,

```
ViewName: REPORTT
Title: #        FieldName             Off             Len             Fmt
Field: 00000 Cro_id                        00000     00008      C
Field: 00001 CompanyName     00008     00100      C
Field: 00002 Address1                  00108     00050      C
Field: 00003 Address2                  00158     00050      C
Field: 00004 Address3                  00208     00050      C
Field: 00005 Address4                     00258     00050      C
Field: 00006 Address5                     00308     00050      C
Field: 00007 Postcode                  00358     00008      C
Field: 00008 Company_ID          00366     00010      C
Field: 00009 Address6                  00376     00015      C
Field: 00010 Telephone              00391     00013      C
Field: 00011 Suspension_Flag 00404     00015      C
Field: 00012 MAX-SCORE                          00419     00003      R
Field: 00013 CLUSTER                            00422     00008      X
Field: 00014 ATTRIBUTES                    00430     00008      X
Field: 00015 00011 C
```

DupFinder

`DupFinder` is a search client that detects duplicate IDT records.

`DupFinder` reads the IDT and treats each record as a search record. Output is written to an external flat file.

The duplicate finding process must use a suitably defined Multi-Search Definition (see *DESIGN GUIDE*) and can process all, or a subset of the IDT records. Limiting the number of records is accomplished by specifying (in the definition) a maximum number of records to process, or by specifying a starting `Record-Id`.

The output by default includes the search record and its duplicates. An option can be specified to limit the output to the search records (which found duplicates) only, or a set of duplicates with the search record removed.

Starting from the Console

The Batch Search Client, `DupFinder`, is started from the Identity Resolution Console Client by selecting **Tools** > **DupFinder**.

CHAPTER 5

Table Loader

This chapter includes the following topics:

- Concepts, 75
- Starting, 75
- Restarting, 77
- Performance, 78
- Fault Tolerance - Data Errors, 80
- Locales, 81

Concepts

The IIR Table Loader extracts data from either a flat-file or database tables and creates an Identity Table (IDT) and Identity Indexes (IDXs).

It is a multi-threaded application and performs the following tasks in parallel:

- Reads input source (flat-file or database)
- Generates keys (multiple threads)
- Sorts and writes output files for DBMS mass load utilities
- Runs DBMS mass load utilities (multiple threads)

The Loader takes checkpoints between phases in its processing and can be restarted after a failure.

Starting

Console

To start the Table Loader, click the **System** > **Load IDT**. Select a **Loader-Definition** from the drop-down list. Progress messages from the Loader will appear in a new window.

The **Stop** button in the Progress Window is used to instruct the Table Loader to abort processing. It may not stop immediately if it is currently running an external process such as the DBMS load utility. The Table Loader does not kill the utility; it waits for it to complete before stopping.

Batch

The Table Loader utility is called `loadit`. It is launched and managed in batch mode using the idsbatch utility.

The idsbatch is used to run user-defined jobs. The available jobs are defined in the User-Job-Definition section of the SDF.

For more information about user-defined jobs, refer to the *User-Job-Definition* and *User-Step-Definition* sections in the *DESIGN GUIDE*. For more information about the `idsbatch` utility, refer to the *Batch Utilities / idsbatch* section in this guide.

Starting Table Loader

To start the IDT load, along with the regular parameters, create an input text file (which has the instructions to perform the IDT load job) and pass it to the idsbatch utility. For example,

```
idsbatch -h%SSA_CSHOST% -id:\idt_load.txt -1d:\idt_load.log -2d:\idt_load.err
```

Instructions in the `idt_load.txt` are,

```
# Run user job
# ------------
action=job-run
job-name=user-job-loadit-IDT
system-name=ssa
rulebase-name=#SSA_RBNAME#
work-directory=#SSAWORKDIR#
```

The job-name (`user-job-loadit-IDT` in the above example) should be defined in the User-Job-Definition section of the SDF. For example,

```
loader-definition
*================
NAME=load-IDT
JOB-LIST=job-loadit-IDT
job-definition
*=============
NAME=job-loadit-IDT
FILE=lf-srn-student
IDT=IDT280
logical-file-definition
*======================
NAME=lf-srn-student
COMMENT="Read from SRN User Source Tables"
PHYSICAL-FILE=IDT280
FORMAT=SQL
user-job-definition
*==================
COMMENT="Load SRN_STUDENT IDT"
NAME=user-job-loadit-IDT
user-step-definition
*===================
JOB=user-job-loadit-IDT
NUMBER=0
NAME=run-loadit
TYPE="Load ID Table"
PARAMETERS=("Loader Definition",load-IDT)
```

Restarting from the beginning

If you wish to completely restart the load from the beginning, you must first Refresh the System. This can be done using the Console Client, or by defining a User-Job and running it with `idsbatch`. This will remove the IDT, IDXs, and any restart information left from the previous load attempt. After this, start the load again as documented above.

DBMS Mass Load Utility Name

The Table Loader will by default use the DBMS load utility that was specified at Install time and is available to it through the environment variable *SSASQLLDR*.

This specification can be overridden or redefined using `DATABASE-OPTIONS=IDTLOAD()`. Refer to the *System Definition* section of the *DESIGN GUIDE* for details.

Restarting

Checkpoints

The Table Loader takes checkpoints after the following major points in its processing:

- Creation of IDT
- Opening data source and creation of triggers (if necessary)
- Source data extraction and key generation
- Load of IDT
- Creation of indexes on IDT
- Analysis of IDT
- Creation, loading and analysis of each IDX

Failures that can be recovered

The Table Loader can be restarted from the last successful checkpoint after a failure, as long as the reason for the failure has been corrected and it does not change the workflow performed by the Table Loader. For example, if the Table Loader:

- ran out of room in the database and more space has been added, or
- the process was cancelled.

the Table loader can be restarted after the last successful checkpoint.

However, anything that changes the workflow performed, or the output generated by the Table Loader invalidates the checkpoint information. For example, if you change

- the data source, or
- Key-Logic and/or options, or
- the number of indexes, or
- Loader options such as Load-All-Indexes, Re-Index, IDT-Only, etc., or
- Sync-level,

the previous checkpoints are invalid and you must not restart the load.

Note: The **Load** button in the System Editor deletes all restart information. Do not edit a System when a restart is pending.

How to Restart

The IIR Console keeps track of the Table Loader's status. When you click **System** > **Load IDT**, the console will offer to restart a load if it failed last time.

To re-start, click **Yes**. If you wish to restart from the beginning, the partially loaded IDT and IDX must first be deleted by running **System** > **Refresh**, followed by running **System** > **Load IDT**.

Performance

The Table Loader uses multiple threads to overlap its work. Multiple threads are used during the data extraction, key generation and DBMS load phases:

Reader

Reads source records from the database or input file and places them in a queue for the Key Generation threads to process.

Key Generation

Processes the source records to create IDX rows. There are n key generation threads by default, where n in the number of CPUs on the machine.

Writer

Writes the IDT and IDX rows to operating system files. These files are used as input to the DBMS Load utility. IDT rows are written directly to a flat-file. IDX rows are pushed into the IIR Sort utility where they are sorted and written to an operating system file.

Loader

Threads merge sort files and run the DBMS load utilities to load the IDT and IDXs in parallel. There are m Loader threads by default, where m in the number of CPUs on the machine.

The Table Loader can be tuned by setting the size of the Reader's input queue and the Writer's sort buffers as well as the number of key generation and loader threads.

Input Queue

The size of the Reader's input queue is set with the environment variable, `SSALDR_RBSIZE=nnn`

where `nnn` is the number of records. The default value is 5000.

This parameter is also used to calculate the size of the key generation output queues. They are calculated as

```
SSALDR_RBSIZE / number_of_key_threads * 8
```

In order to keep the Key Generation and Writer threads busy, the input queue must be filled as quickly as possible.

Flat-File Input

When reading from a flat-file, the input queue can be filled very quickly, and in general, the bottleneck is in the Key Generation and/or Writer threads. Since the Writer thread blocks for a short period during sort processing, it is advantageous to have a large input queue (and therefore large key generation output queues), so that key generation can proceed concurrently.

Database Input

When the Reader's input queue is filled from records from a database, the Reader thread is usually the bottleneck and the other threads spend time waiting for work.

Finding the bottleneck

A thread can wait for two reasons:

- waiting for work in its input queue, or
- waiting for space in its output queue (where it places its results)

To determine how often a thread had to wait, refer to the statistics in the Table Loader log file. When each thread ends, it reports the number of times it had to wait for work. For example,

```
Reader thread [1] ends. Records In 900000. Waits 960
Keygen thread [3] ends. Processed 450000. Waits 214
Keygen thread [4] ends. Processed 450000. Waits 214
Writer thread [2] Extract ends. IDT out 900000 Waits 448
```

The thread with the least number of "waits" is the busiest thread (bottleneck). In the example above, the Key Generation threads were the busiest. The Reader thread spent some time waiting for the Key threads to make room in Reader's output queue. This is typical of a flat-file load.

When reading from a database, it in not uncommon for the Reader thread to report zero waits. That is, it was reading records as fast as the DBMS could deliver them and the other threads were able to keep up with the work load by keeping the input queue in a state where there was always enough room to add the incoming records.

Tuning

The objective is to make the input queue large enough to keep it from becoming the bottleneck.

If reading from the database and the reader thread reports 0 waits, the reader queue is long enough. If reading from a flat-file, the reader queue must be set large enough so that the key generation threads are the busiest (least waits).

Sort Buffers

The Writer thread takes records from the Key Generation output queues and passes them to the IIR Sort routine. The sort places each row into a memory buffer. When the buffer becomes full, its contents are sorted and the results are written to a sort work file on disk. Once all groups are sorted, the groups on disk are merged to create a fully sorted file. The fully sorted file is used as input to the `DBMS Load` utility.

The performance of the sort is affected by the

- size of the sort buffer,
- number of sort threads, and
- the placement of the disk files.

These are controlled by the DATABASE-OPTIONS=IDXSORT (. . .) parameters defined in the SDF. The default sort buffer size is 64MB.

A large sort buffer is desirable because

- there will be less sorted groups to merge (less random I/O)
- sorted groups are written in bursts of I/O, so they create less disk contention
- they allow larger I/O buffers during the merge phase

However large sort buffers will hold more unsorted records, and therefore they will be sorted less often and each sort operation will take longer (as compared to a smaller sort buffer).

While sorting occurs, the writer thread is blocked. This means it can not remove records from the key generation output queues, so they in turn will block if there is insufficient room to write their results. Therefore it is important that the key generation output queue is large enough to enable key generation to continue while sorting occurs. Since the key generation output queue size is determined by `SSALDR_RBSIZE`, it must be set quite high when large sort buffers are in use.

Tuning

Allocate as much sort memory as possible. Make sure it is not so large as to cause swapping to occur, as this negates the benefit of a memory based sort.

If the Key Generation threads have more waits than the writer thread, it indicates that `SSALDR_RBSIZE` should be increased.

Place the sort work file on a different device to the output file to avoid disk contention.

Compress-Key-Data

The appropriate size of the Compress-Key-Data parameter must be determined. Load a representative sample of data and use the histkg utility to determine the appropriate setting. Refer to the *Compressed Key Data* section in the *DESIGN GUIDE* for details.

DBMS Extents

When loading large amounts of data, it is wise to allocate large extents for the IDT and IDXs. Use the `DATABASE-OPTIONS=IDT(. . . )` and `IDX(. . . )` to allocate large extents and/or place the tables and indexes in appropriate tablespaces.

Partitioning Data

Loading extremely large systems requires a scalable solution. In this situation, consider partitioning the data on a logical criterion such as a range of IDs. Create one system per partition and load them in parallel.

CPU and I/O usage

Key Threads

The Table Loader automatically creates n key generation threads, where n is the number of CPUs available. You may override this value by setting the environment variable *SSALDR_KEYTH=n*.

Loader Threads

The number of Loader threads is set to the number of CPUs available on the machine. You may override this value by setting the environment variable *SSALDR_LOADTH=n*.

In general, the DBMS load is an I/O intensive operation. Creating too many Loader threads may cause I/O contention that could slow down the load process. Not all loader threads can be used in some cases:

1. When there is insufficient work to utilize all threads.
2. When there are only Lite Indexes left to load and the IDT has not been loaded yet.
3. When loading to a UDB database, UDB creates tablespace locks that prevent concurrent loads.
4. When loading to MSQ, all merge phases must be completed prior to starting the first mass load utilit (`bcp`).

Fault Tolerance - Data Errors

The Table Loader will terminate with an error if the DBMS Load utility reports any errors while loading the IDT or IDXs. This may be undesirable if the failure is caused by a small number of data errors in the source rows.

The `DATABASE-OPTIONS=IDTERR` is used to specify the maximum number of data errors that can occur before the Table Loader will report a fatal error. The default value is zero.

Note: Allowing data errors may produce integrity errors in the IDT and/or IDXs. The exact nature of the integrity error is database dependent:

- Oracle's SQL*Loader and UDB's LOAD utility reject erroneous rows and writes them to an error file, so they will be missing from the IDT. However, these rows will still be present in the IDX, since IDX rows are stored in binary form and are only interpreted by IIR.

- UDB's Import utility does not reject rows. Instead, the rows are loaded to the IDT with the incorrect column values set to NULL.

Correcting Errors

The source data should be corrected and the IDT/IDXs reloaded. If the System is synchronized the Update Synchronizer will automatically correct the IDT and IDXs while processing UST updates. Therefore it is unnecessary to reload the tables.

Locales

The Table Loader parses the DBMS's Load Utility log file in order to determine if the load succeeded. By default it searches for English language phrases in the log file. However, when the database server is installed on a machine that uses a non-English locale, the DBMS Load Utility will write its log file using that character set. In these circumstances, special environment variables must be defined to specify replacement phrases to search for. Failure to do so will result in erroneous load failure messages reported by the Table Loader.

Oracle

The IIR Table Loader (loadit) checks the number of records loaded by SQL*Loader. To do this, loadit parses the text of SQL*Loader's output looking for particular strings. These strings are expected to be in English.

When a foreign language version of Oracle is used, two environment variables must be defined to specify the foreign language text that corresponds to the English strings that loadit is looking for.

Set the environment variables *SSALDR_ORA_READ_TXT* and *SSALDR_ORA_REJECT_TXT* to the foreign language strings that correspond to "Total logical records read:" and "Total logical records rejected:" messages respectively. These variables must be the complete string, up to and including the ':', starting from the left margin of the output.

Example: An extract from a SQL*Loader Log in English:

```
Space allocated for bind array:21248 bytes(64 rows)
Space allocated for memory besides bind array:0 bytes
Total logical records skipped:                          0
Total logical records read:                             6
Total logical records rejected:                         0
Total logical records discarded:                    0
```

Example: An extract from a SQL*Loader log using a non-English locale:

```
Ba_lama dizisi i_in tahsis edilen bo_luk:21248 byte(64 satr)
Bellek i_in ba_lama dizisinin d_nda tahsis edilen bo_luk: 0 bytes
Toplam atlanan mantksal kayt:                       0
Toplam okunan mantksal kayt:                           6
Toplam edilmeyen mantksal kayt:                     0
Toplam atlan mantksal kayt:                           0
```

In this case, set the environment variables to the following:

```
set SSALDR_ORA_READ_TXT=Toplam okunan mantksal kayt:
set SSALDR_ORA_REJECT_TXT=Toplam edilmeyen mantksal kayt:
```

MSQ

The MSQ implementation of the Table Loader searches for the phrase `" rows copied."` as this precedes the number of rows loaded into the table. For example, 10000 rows were loaded in the follow example:

```
Starting copy...
1000 rows sent to SQL Server. Total sent: 1000
1000 rows sent to SQL Server. Total sent: 2000
1000 rows sent to SQL Server. Total sent: 3000
1000 rows sent to SQL Server. Total sent: 4000
1000 rows sent to SQL Server. Total sent: 5000
1000 rows sent to SQL Server. Total sent: 6000
1000 rows sent to SQL Server. Total sent: 7000
1000 rows sent to SQL Server. Total sent: 8000
1000 rows sent to SQL Server. Total sent: 9000
1000 rows sent to SQL Server. Total sent: 10000
10000 rows copied.
Network packet size (bytes): 4096
Clock Time (ms.): total 640 Avg 0 (15625.00 rows per sec.)
```

When using a non-English locale, you may provide alternate text for this phase using the server environment variable *SSALDR_MSQ_COPIED_TEXT*.

CHAPTER 6

Update Synchronizer

This chapter includes the following topics:

- Overview, 83
- updsync utility, 86
- updmulti utility, 90
- Restarting Automatically, 94
- Synchronization Level, 94
- Transaction File / Table, 96
- Integrity Checking, 99
- Performance, 99
- Timing Window, 101

Overview

The Update Synchronizer is a background process that applies updates to Identity Resolution Tables and Indexes to keep them synchronized with changes to User Source Data. It can also compare the contents of the IDT/IDX against the User Source Data and report any differences.

IDTs created with the SYNC option can be synchronized with User Source Data.

User Source Data

User Source Data is held in an SQL database but does not have to be. It might be loaded into Identity Resolution from a sequential file (known as a Flat-File in Identity Resolution terminology).

When the User Source Data is held in a database that Identity Resolution can directly access using SQL, the data are said to reside in User Source Tables (UST).

Synchronization against a UST includes the following tasks:

- reading transactions from an SQL accessible table known as the Transaction Table.
- accessing User Source Data with SQL
- applying updates to the IDTs and IDXs

Synchronization against a Flat-File source includes the following tasks:

- reading transactions from a Flat-File
- applying updates to the IDTs and IDXs

Supplying Transactions

Transaction Data might be read from a Transaction Table or from a Flat-File.

Source Access - Transaction Table

1. A Transaction Table is an SQL accessible table named `IDS_UPD_SYNC_TXN` held in the source database. It holds information about inserts, updates, and deletions from USTs.
2. The information in the table records the order in which these events occurred, together with primary key values of the affected source rows.
3. Identity Resolution permits the separation of USTs and IDTs on different databases. All updates to USTs are logged in a Transaction Table residing in the source database to prevent distributed database updates when source rows are modified.

Transactions can be added to the table in two ways:

Transactions added by Triggers

By default, database triggers are attached to the USTs by the Table Loader before source extraction. The triggers automatically insert transactions into the Transaction Table when UST updates occur. Triggers are a reliable method of transaction creation because the DBMS ensures that triggers are fired whenever updates occur.

Note: Most databases do not fire triggers when the source table is maintained by using a mass-load utility. This results in a loss of synchronization.

Oracle does not fire triggers under certain circumstances, such as the addition of records to the source tables by SQL*Loader when using the `DIRECT-PATH` facility.

Microsoft SQL Server: If you are using Microsoft SQL Server DTS to bulk load records, clear the **Use Fast Load** option (enabled by default) under the **Options** tab of the Data Transformation Task property sheet. If the operation is performed using bcp's `BULK INSERT` statement, specify the FIRE_TRIGGERS options.

Transactions added Manually

Some OEM developers prefer not to rely on triggers. Instead they want to directly insert transactions into the Transaction Table at suitable points in their application logic. The disabling of trigger creation is achieved by setting the `Txn-Source` clause to a value of `MANUAL`. The creation of valid transactions then becomes the user's responsibility.

As an aid, the Table Loader still generates trigger code but instead of attaching the triggers to the USTs, it writes their source code to the Table Loader's log file. The user must perform the equivalent actions as the trigger code when inserting transactions into the Transaction Table. Any deviation from the order of transaction creation or content will result in incorrect synchronization results.

Note: Informatica Corporation reserves the right to change the trigger format / content at any time. Using the manual trigger option exposes you to the possibility that you might have to change your code. Some degree of independence is afforded by not directly inserting transactions into the transaction table. Instead, call the **IDS_UPDATE_SYNC** package to do this (as the automatically generated trigger code does). The trigger code gathers the required data and passes it to the package for formatting and insertion into the Transaction Table.

No Source Access (NSA)

When access to the source database is not possible, the synchronization method is known as No Source Access. In this situation, the transaction data must contain all the information required to add or delete records from the IDT without referring to any source data. In other words, the transactions must contain complete IDT records. They can be read from either an operating system file known as the Flat-File or from a database table (NSA Transaction Table).

Flat-File

A "flat-file" contains records in IDT format so that the Synchronizer can directly add (delete) them to (from) the IDT. Of course, the Synchronizer also updates the IDXs to reflect the changes there as well.

If you plan to synchronize using flat files the UST must be sourced from a flat file as well. See the `sourced_from` clause in the *DESIGN GUIDE* for the appropriate syntax. See the Transaction File section for more details about the Flat-File layout.

NSA Transaction Table

There is an alternative to providing IDT rows in a Flat-File. The Synchronizer can also read transactions from an SQL table known as the NSA Transaction Table. It is similar in content to a Flat-File. However, it has the advantage that it does not need to be "closed" before passing it to the Synchronizer for processing. See the Transaction File section for more details about the NSA Transaction Table.

Synchronizer Process

The Update Synchronizer process updates the IDT database. At startup, it connects to one of the following components:

- All source databases used by the specified System
- A flat transaction file specified by the -f parameter
- The target database (when using an NSA TransactionTable)

It periodically polls for work by reading the transaction table on each source database. This is known as a duty cycle. A duty cycle can begin in one of two ways:

- a specified period of time has elapsed since the last duty cycle (`-t`parameter), or
- a new duty cycle commences immediately (without sleeping) if the previous cycle processed any transactions.

It processes a maximum of `Rate` transactions for each duty cycle for each source database before committing the results. The default `Rate` of 100 can be changed using the `-m` parameter. This prevents any one source database from monopolizing all of the Synchronizer's time at the expense of less active source databases.

If the only source is a flat transaction file, the Synchronizer shuts down automatically when it reaches EOF.

Although designed to be a near real-time process, delays in synchronization are possible for multiple reasons:

- USTs are updated while the IDT is still being loaded (that is, the IIR-ID Table and Indexes do not exist yet)
- the USTs and IDTs are on different databases and the network link is down.
- the Synchronizer process is not running while updates occur.

In these situations, any updates to the USTs are logged and reapplied at a later stage (when using a Transaction Table).

Synchronizer Utilities

You can use the following Update Synchronizer utilities:

updmulti

You can use the updmulti utility to synchronize with an IDT in the following scenarios:

- If the IDT uses triggers as the transaction source
- If you apply updates to the IDT by using the Real Time API or the Real Time Web Service

The updmulti utility improves the synchronizer performance when it handles many IDT updates.

updsync

The updsync utility is deprecated, and Informatica recommends that you use the updmulti utility to synchronize IDTs.

updsync utility

The updsync is named as `Update Synchronizer` utility. This section provides information on how to start and stop this utility.

Starting updsync

Start the updsync utility from the Console Client, use **Tools** > **Synchronizer** or start from the command line.

If you start it from the command line, be sure to specify the -5 switch to enable communication and control facilities from the Console.

The command uses the following syntax:

On Windows:

```
%SSABIN%\updsync -r<Rulebase> -p<System> -h<Rulebase Server Host>:<Port> [Optional Switches]
```

On UNIX:

```
$SSABIN/updsync -r<Rulebase> -p<System> -h<Rulebase Server Host>:<Port> [Optional Switches]
```

where

Rulebase

Rulebase connection string. Use one of the following formats for the rulebase connection string:

- `odb:0:userid/password@service`
- `iir:rb`

System

Name of the system to synchronize.

Rulebase Server Host:Port

Host name and port of the Rulebase Server.

If you use Rulebase Server Groups, you can replace this parameter with the `-g<Rulebase Server Group Name>,<Rulebase Connection String>` parameter.

If you use Apache ZooKeeper, you can replace this parameter with the `-z<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>]` parameter.

Optional Switches

The following parameters are supported:

-cMaxCycles

Specifies the maximum number of duty cycles to run before shutting down. The default is to run until instructed to shut down, see *Stopping updsync*.

-eIDT

Specifies that only transactions that affect the specified IDT will be processed. This permits the synchronization of a single IDT when multiple IDTs have been defined in the System. The default (when –e has not been specified) is to synchronize all IDTs in the System.

-fFlatFile

The name of the transaction file when using flat file synchronization.

-g<Rulebase Server Group Name>,<Rulebase Connection String>

Name of the Rulebase Server Group and the rulebase connection string.

-i[IDT[,IDX]]

Check the integrity of all IDTs and IDXs in this system, or a particular IDT and IDX. See the *Integrity Checking* section for details.

-k

Display erroneous records in detail. Used in conjunction with -i.

-l

Assume case of system name in txn file/table matches the case of the system name specified with the -p parameter. When not specified, a case insensitive (more expensive) select and compare mechanism is used. Transactions stored by triggers in the txn table insert the system name in lower case.

-mRate

Commit rate (defaults to 100).

-n

Treats the transaction file as a text file where records are separated by a newline. Without this option, the transaction file is interpreted as a binary file.

-oTime

Collect Optimizer statistics every `Time` seconds.

--parallel=n

Optional. Applicable only for Oracle. Indicates the number of threads that you want to use for the synchronization process. Use this parameter to perform parallel processing for improved performance.

-tTimeOut

Specifies the number of seconds between duty cycles. The default value is 60 seconds. A value suffixed with ms is treated as milliseconds.

-vpsui

Verbosity (p=progress, s=stats, u=usage, i=info)

-yMax[,Wait]

Fault tolerance feature. Synchronizer automatically restarts Max times in case of failure. For more information, see the Restarting Automatically section.

-zTxn

Transaction sequence number to delete.

-z<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>]

List of ZooKeeper servers based on the host name of the ZooKeeper server and the port on which the host listens for connections.

-5<Job Number>:<Console Server Host Name>:<Console Server Port Number>

Required when you use flat-file synchronization. Indicates that the specified job connects to the Console Server through the specified port number. Use this parameter to enable the progress messages and the ability to shut down the synchronizer from the console. Specify the job number as 0 to connect to the Console Server and register the job as an anonymous job. Other job numbers are reserved for the jobs that the console launches.

--rbcheck

The Update Synchronizer periodically checks its communication channel to the Rulebase Server. Use `--rbcheck` to stop the Update Synchronizer when the Rulebase Server stops with a hard shutdown.

The -d option specifies the time duration to retry the connection to the Rulebase Server. If the Informatica Identity Resolution system exceeds the time duration to retrieve connectivity, the Update Synchronizer exits the services.

When you start the the Update Synchronizer with –rbcheck, -d, and -y options, the -d option overrides the -y option in the case of rule base check failure.

--validate

Validates data when you synchronize the data with Identity Table (IDT) using a flat file or NSA table.

By default, the Update Synchronizer performs the following validation checks:

Field	Validation
Numeric - N	Checks for a numeric value that aligns to the right and has leading zeros instead of spaces.
Character String - C	Checks for spaces. Does not allow a null value (0x0000) as a padding character.
Unicode String - W	Checks if the Unicode spaces (0x0020) has padding. Does not allow a null value (0x0000) as a padding character.

Use the `--validate` option to validate the data. The following validations are optional as the errors calculated here are based on percentage of occurrence:

Field	Validation
Character String - C	Counts the number of rows where values in column are using full buffer. Reports an error when 99% of data meets this condition. This may not be an error and could be due to data truncation.
Unicode String - W	Counts the number of rows where values in column are using full buffer. Reports error when 99% of data meets this condition. This might not be an error and can be due to data truncation. Count the number of rows where values contains invalid Unicode spaces as padding character, that is mix of endianess 0x2000 and 0x0020. The problem is 0x2000 is also valid Unicode, so if the 75% or more rows are ending with 0x2000 character on a big endian system or 0x0020 on little endian system, then report it as an error. Count the number of rows where values contains ASCII spaces instead of Unicode spaces as padding character, that is 0x2020 is used instead of 0x0020. Again the problem is 0x2020 is also a valid Unicode, so if the 75% or more rows are ending with 0x2020 character then report it as an error.

Output goes to the console, and you can redirect it to a log file if required. On Windows, run the following command to start the updsync utility:

```
start /min %SSABIN%\updsync -rodb:0:userid/password@service -p<System> -vp
-h%SSA_RBHOST%
```

The `System` parameter indicates the name of the system that you want to synchronize, and the `odb:0:userid/password@service` string indicates the rulebase connection string. You can also use `iir:rb` as the rulebase connection string, which does not expose the password on the command line.

Stopping updsync

You can stop this utility using the console or through script.

Via Console

You can shut down the Update Synchronizer if you started it from the Console or from a command line with the -5 switch.

The Console sends a message to the Synchronizer to stop when the next duty cycle begins. The Synchronizer will acknowledge the receipt of the shutdown request by displaying a progress message and will shutdown in due course.

Via Script

Alternatively, you may schedule the Synchronizer to shut down by running the following script to add a Shutdown Request to the transaction file. It will not shut down until it processes the request. This may take a while if there is a backlog of transactions to process. Therefore it is recommended that the Synchronizer be shut down via the Console.

For Win32:

```
%SSABIN%\syncstop System Uid Pwd Svc [DBType]
```

For Unix:

```
$SSABIN/syncstop System Uid Pwd Svc [DBType]
```

where

System

The name of the System being synchronized.

Uid

The SSA userid defined for UST database.

Pwd

The password for the SSA userid defined for UST database.

Svc

The name of the UST database.

DBType

An optional database type of the UST database specified when the environment variable *SSA_DB_TYPE* is not set. Specify `ora`, `udb`, `myq` or `msq`.

The Update Synchronizer will shut down when the next duty cycle begins.

UDB/DB2: The Win32 syncstop script must be run from a DB2 Command Window.

Note: The syncstop script cannot be used to stop a Synchronizer that is processing transactions from the NSA table. The only supported mechanism in this case is to use the **Stop** button on the Console, or the --rbcheck switch.

Server Shutdown

Identity Resolution servers do not shut down if the clients are attached, unless you use the hard shutdown option.

updmulti utility

This section provides on how to use the updmulti utility.

Prerequisites

updmulti is a client of the Real Time Web Service. Therefore, the Synchronization Server must be running and the Real Time Web Service must be configured appropriately for the IDT to be synchronized. See the *Enabling the Real Time Web Service* for details.

Starting updmulti

You can start updmulti from the Console Client by selecting **Tools** > **Synchronizer**.

Alternatively, you can launch it from the command line. If you use the command line method, be sure to specify the -5 switch to enable communication and control facilities from the Console.

The command uses the following syntax:

On Windows:

```
%SSABIN%\updmulti -r<Rulebase> -p<System> -h<Rulebase Server Host>:<Port> -e<IDT>
 [Optional Switches]
```

On UNIX:

```
$SSABIN/updmulti -r<Rulebase> -p<System> -h<Rulebase Server Host>:<Port> -e<IDT>
 [Optional Switches]
```

where

Rulebase

Rulebase connection string. Use one of the following formats for the rulebase connection string:

- `odb:0:userid/password@service`
- `iir:rb`

System

Name of the system to synchronize.

IDT

Specifies the IDT that you want to process. Ensure that the IDT is available in the specified system.

Rulebase Server Host:Port

Host name and port of the Rulebase Server.

If you use Rulebase Server Groups, you can replace this parameter with the `-g<Rulebase Server Group Name>,<Rulebase Connection String>` parameter.

If you use Apache ZooKeeper, you can replace this parameter with the `-z<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>]` parameter.

Optional Switches

The following parameters are supported:

-cMaxCycles

Specifies the maximum number of duty cycles to run before shutting down. Not relevant for Flat-File input. The default is to run until instructed to shut down (see *Stopping updmulti*).

-fFlatFile

The name of the transaction file when using flat file synchronization.

-g<Rulebase Server Group Name>,<Rulebase Connection String>

Name of the Rulebase Server Group and the rulebase connection string.

-z<Host Name1>:<Port1>[,<Host Name2>:<Port2>,<Host Name3>:<Port3>…, <Host NameN>:<PortN>]

List of ZooKeeper servers based on the host name of the ZooKeeper server and the port on which the host listens for connections.

-i[IDT[,IDX]]

Check the integrity of all IDTs and IDXs in this system, or a particular IDT and IDX. See the *Integrity Checking* section for details.

-k

Display erroneous records in detail. Used in conjunction with -i.

-l

Assume case of system name in txn file/table matches the case of the system name specified with the -p parameter. When not specified, a case insensitive (more expensive) select and compare mechanism is used.

-mRate

Commit rate (defaults to 100). This parameter is relevant for flat-file synchronization only. Determines how frequently 2 phase commit records are saved. To set the commit rate for Synchronization Server, add the `txn_commit_rate` parameter to its configuration file. For more information, see "Configuration Settings" on page 149.

-n

Treats the transaction file as a text file where records are separated by a newline. Without this option, the transaction file is interpreted as a binary file.

-oTime

Collect Optimizer statistics every Time seconds.

-Sn

Disconnects the connection to the source database and reconnects to it after the specified number of duty cycles.

-tTimeOut

Specifies the number of seconds between duty cycles. The default value is 60 seconds. A value suffixed with ms is treated as milliseconds.

-vpsui

Verbosity (p=progress, s=stats, u=usage, i=info)

-yMax[,Wait]

Fault tolerance feature. Synchronizer automatically restarts Max times in case of failure. For more information, see the Restarting Automatically section.

-zTxn

Transaction sequence number to skip.

-ZTxn

Transaction sequence number to delete.

-5<Job Number>:<Console Server Host Name>:<Console Server Port Number>

Required when you use flat-file synchronization. Indicates that the specified job connects to the Console Server through the specified port number. Use this parameter to enable the progress messages and the ability to shut down the synchronizer from the console. Specify the job number as 0 to connect to the Console Server and register the job as an anonymous job. Other job numbers are reserved for the jobs that the console launches.

--offset=nnn

This is an optional parameter which applies to flat-file processing only. This number, if present, is added to the sequence number of each record processed. May be used to ensure global uniqueness of flat-file transactions.

--parallel=n

Optional. Applicable only for Oracle. Indicates the number of threads that you want to use for the synchronization process. Use this parameter to perform parallel processing for improved performance.

--rbcheck

Requests periodic checks of Rulebase Server connectivity and to abort when inaccessible. Under normal circumstances, updmulti accesses the RB Server very seldom, and is therefore unaware that it may have stopped. This option is useful to automatically stop updmulti when the servers have been stopped with a hard shut down.

--validate

When synchronizing data using flat file or NSA table to Identity Table.

By default, **updmulti** performs the following validation checks:

Field	Validation
Numeric - N	Checks for a numeric value that aligns to the right and has leading zeros instead of spaces.
Character String - C	Checks for spaces. Does not allow a null value (0x0000) as a padding character.
Unicode String - W	Checks if the Unicode spaces (0x0020) has padding. Does not allow a null value (0x0000) as a padding character.

Use the `--validate` option to validate the data. The following validations are optional as the errors calculated here are based on percentage of occurrence:

Field	Validation
Character String - C	Counts the number of rows where values in column are using full buffer. Reports an error when 99% of data meets this condition. This may not be an error and could be due to data truncation.
Unicode String - W	Counts the number of rows where values in column are using full buffer. Reports error when 99% of data meets this condition. This may not be an error and could be due to data truncation. Count the number of rows where values contains invalid Unicode spaces as padding character, that is mix of endianess 0x2000 and 0x0020. The problem is 0x2000 is also valid Unicode, so if the 75% or more rows are ending with 0x2000 character on a big endian system or 0x0020 on little endian system, then report it as an error. Count the number of rows where values contains ASCII spaces instead of Unicode spaces as padding character, that is 0x2020 is used instead of 0x0020. Again the problem is 0x2020 is also a valid Unicode, so if the 75% or more rows are ending with 0x2020 character then report it as an error.

When you use the `--validate` option and it results in errors, the data cannot not be rolled back. It is recommended to use this option in a test environment. Ensure that the input data has correct Unicode spaces.

--reader_sz=nnn

Size of reader circular buffer size. The default value is 5000.

Output goes to the console, and you can redirect it to a log file if required. On Windows, run the following command to start the updmulti utility:

```
start /min %SSABIN%\updmulti -rodb:0:userid/password@service -p<System> -vp
-h%SSA_RBHOST%
```

The `System` parameter indicates the name of the system that you want to synchronize, and the `odb:0:userid/password@service` string indicates the rulebase connection string. You can also use `iir:rb` as the rulebase connection string, which does not expose the password on the command line.

Stopping updmulti

This section describes about how to stop the updmulti utility.

Via Console

updmulti can be shut down from the Console if it was started from either

- the Console, or
- from a command prompt and the -5 switch was specified.

The Console sends a message to updmulti to c`"stop when the next duty cycle begins".` updmulti will acknowledge the receipt of the shutdown request by displaying a progress message and will shutdown in due course.

Server Shutdown

The synchronizer periodically checks its communication channel to the Rulebase Server when started with the `--rbcheck` switch. If the Rulebase Server stops for any reason (for example, due to a hard shutdown), the Update Synchronizer terminates with an error condition.

Note: Identity Resolution servers do not shut down if the clients are attached, unless you use the hard shutdown option.

Restarting Automatically

The Synchronizer has the ability to restart itself automatically in case of failure. This feature should be used carefully as it is undesirable to attempt a restart when the previous failure was caused by as a non-transient error, such as a database instance failure or a Tablespace running out of room.

Automatic restarts are enabled using the -y switch:

```
-yMax[,Wait]
```

where

Max

This is a positive number and represents the maximum number of restart attempts. A value of zero is treated as "unlimited". The default value is 100.

Wait

This is optional and represents the number of seconds to wait before attempting a restart. This can be used as a throttling mechanism to prevent many restart attempts in quick succession. The default value is 5.

We recommend the values `-y100,5`. If the Update Synchronizer fails to restart after the specified number of retries, the error is unlikely to be transient and requires investigation and correction.

Synchronization Level

The Synchronizer operates most efficiently when the nominated primary keys (PKn notation) are guaranteed to be unique. It is advisable that the User Source Tables are defined with integrity constraints to ensure this fact. However, in cases where this is not possible, a Synchronization Level can be specified which allows the Synchronizer to tolerate, or even expect duplicates.

The following synchronization levels may be specified:

Synchronization Level	Check on Load	Check on Sync	On Sync Error. . .
REJECT_DUPLICATE_PK	Yes	Yes	Stop / Ignore
REPLACE_DUPLICATE_PK	Yes	Yes (NSA only)	Replace row
WARN_DUPLICATE_PK	No	Yes	Issue Warning
ALLOW_DUPLICATE_PK	No	No	N/A

The default level is `REJECT_DUPLICATE_PK`. This specifies that an IDT cannot be loaded when duplicates are present. Rows containing duplicate PKs will not be added to the IDT, but may exist on the UST (where the Synchronizer has no control over them). Customers requiring duplicate protection of their source tables should use database constraints to prevent creating duplicates. The synchronization process operates most efficiently in this mode.

If only a few duplicates are present and/or you do not want the Synchronizer to stop when a duplicate is detected, use `WARN_DUPLICATE_PK`. This setting will process transactions less efficiently than `REJECT_DUPLICATE_PK` but will produce correct results even when duplicates are present. In cases where a duplicate is detected it will issue a warning and continue.

If the PK is known to be non-unique specify `ALLOW_DUPLICATE_PK`. This informs the Synchronizer to use algorithms that produce correct results when duplicates are present. However, this mode is less efficient than `REJECT_DUPLICATE_PK`. This mode must be used when the PK may contain NULL values.

The synchronization level `REPLACE_DUPLICATE_PK` can only be used in conjunction with an NSA transaction source. An add transaction (type 'A') containing a duplicate PK value will replace the existing IDT row with the new value from the NSA Transaction Table.

Reject_Duplicate_PK

Correct synchronization relies on the ability to uniquely identify User Source PKn Tables records. A User Source Table Definition nominates the source table column(s) that are used to create a unique primary key [with the (`PKn`) notation].

When loading the IDT, the host DBMS will check the PK columns for unique values. The IIR Table Loader will fail with an appropriate error messages if the USTs contain any duplicates. Therefore it is advisable that User Source Tables are defined with constraints to avoid this potential problem.

Without constraints on the columns, it is possible for a user transaction to create a duplicate PK value via an insert or update to a UST. The Synchronizer will detect this situation when attempting to add the same record to the IDT. A new row with a duplicate PK will not be added to the IDT.

If the new "duplicate" row is not identical to an existing row (excluding PK values), the Update Synchronizer reports the situation with an error message. A sample error message is as follows:

```
constraint violated by insert/update to UST
IDT 'IDS_01_IDT01'
UST Key field(s): SSA09.TESTX01A.EMPNO
PK1 '99'
```

Note: Identical duplicate rows are not added to the IDT and not reported as duplicates, as there are some situations where a specific transaction order may produce rows that appear to be duplicates when in fact, they are not.

The example above tells us that a transaction was applied to an IDT called `IDS_01_IDT01`. The PK field for this IDT contains values extracted from a User Source Table column called `SSA09.TESTX01A.EMPNO` and we have attempted to add a duplicate value of `99`.

The Synchronizer will roll back the transaction(s) updated during the current duty cycle and terminate with the above message. Manual intervention is required to

- repair the UST integrity problem (remove the duplicate), and
- inform the Synchronizer to delete the problem transaction.

Repairing the UST

Removing Duplicates

This section is relevant to Synchronization Level `REJECT_DUPLICATE_PK`. The record containing the duplicate primary key must be removed. This is a user responsibility. Once again, this problem could have been prevented if the UST contained DBMS integrity constraints to enforce the uniqueness of the PK column(s).

After deleting the duplicate record, you must update the original (correct) record to force the Synchronizer to re-index the IDT using the values in the correct record. This is necessary because the deletion of the duplicate generates a trigger that will delete all IDT records with this key. The subsequent update will result in the correct record being added to the IDT.

When updating the correct record, be sure to update a column that is present in the IDT, that is a column from the UST that appears in the IDT's `sourced_from` clause. This is required because update triggers are only fired when a `sourced_from` column is modified.

Note: If you do not update the correct record after deleting the duplicate, the IDT will not be correctly synchronized with the UST.

Restarting the Synchronizer

This section is only relevant to Sync Level `REJECT_DUPLICATE_PK`.

It is not possible to simply restart the Synchronizer because the transaction which attempted to add a duplicate is still in the `IDS_UPD_SYNC_TXN` table and will be reprocessed. You must inform the Synchronizer to delete this transaction by using the `-z` switch.

The recommended procedure is to start the Synchronizer with the -m1 parameter. This will commit updates after every successful transaction is processed and rollback/terminate when the "duplicate transaction" is reprocessed leaving it at the head of the transaction queue.

Then start the Synchronizer with the following parameters to delete the "duplicate transaction", commit it and shutdown the Synchronizer:

```
-zTxn -m1 -c1
```

where `Txn` is the transaction sequence number of the failed transaction. You can now restart the Synchronizer with its normal parameters.

Transaction File / Table

In the cases where Identity Resolution can not create triggers on the USTs (for example, when the source database is not supported), or a flat file was used as input to the Table Loader, update transactions must be provided in either a "flat file" or an NSA Transaction Table.

Flat-File Layout

The "flat file" is a binary file containing fixed length records (with no newline separators). It must start with a Control Record. The Control Record is immediately followed Transaction Records. Each Transaction Record consists of a fixed length header followed by an IDT record.

Alternatively, the transaction file can be treated as a text file when the Synchronizer is started with the `-n` switch. In text mode, the Control Record and all Transaction Records must be separated by a newline character. Text mode is only suitable when the IDT records do not contain any binary data that may be confused with a newline. A binary input file is the preferred and safest option.

Control Record

Offset	Length	Description
0	4	Version
4	32	System Name
36	32	Time Stamp
68	64	Reserved for future use

Version

Defines the version of the Control Record. The only valid value is "0001".

System Name

Defines the System that these transactions belong to. Only one System per transaction file is permitted.

Time Stamp

An alphanumeric string containing the date and time when the file was created. The Synchronizer saves this field in its restart information. Format is `"YYYYMMDD HH:MM:SS"` without the quotes.

Reserved

This is not used currently.

Transaction Record

Offset	Length	Description
0	10	Sequence Number
10	1	Operation Code
11	32	IDT Name
43	variable	IDT Record

Sequence Number

The Transaction Sequence Number represented by printable decimal digits. The input file must contain ascending sequence numbers (right justified and zero filled) starting from `0000000001`, without any gaps in the sequence numbers.

Operation Code

Defines the operation to be applied. Valid values are 'A' meaning add this IDT record, and 'D' meaning delete this IDT record.

IDT Name

The name of the IDT that this record belongs to. This is the fully decorated table name as it appears on the target database. For example an IDT named `IDT-99` in the definition file stored on `dbid 01`, would be called `IDS_01_IDT_99`.

IDT Record

The IDT record in the Identity Resolution database format. Fields must be in the same order as defined in the UST-Definition Section of the SDF file. For more information about the field types, see *Informatica Identity Resolution Design Guide*.

Flat-File Rules

The transaction file must be closed by the creating application prior to being used as input to the Synchronizer. The content of the transaction file must not be changed once the update Synchronizer has started using it. Once all transactions have been processed successfully, as verified by inspection of the Update Synchronizer output, the file might be deleted.

The Synchronizer uses the Transaction File Name appended to the System Name to store restart information (like the Time Stamp). When the Synchronizer restarts, it checks the Time Stamp in the Control Record against the stored value. If they differ it reports a "loss of synchronization error" and aborts. This is because the contents of the input file has changed since it was first used.

Restart information is never removed. This is a safeguard against accidental reapplication of the same transaction file. If this were to occur, the Synchronizer will recognize that it has completed processing the file and ignore the transactions.

Since restart information is not removed, Transaction File Names can not be reused. The best approach is to generate file names from the current date and time such that each one is unique. This also helps to identify which one is to be applied next (as transactions must be applied in chronological order).

There is no operation code for an update. Updates are processed using the trigger paradigm. That is, an "update" consists of two transactions; a delete transaction (containing a copy of the IDT record prior to the "update"), followed by an add transaction (containing a copy of the IDT record after the "update").

Special handling is required for non-unique PKs. The Synchronizer will delete all records with the same PK value when processing a Delete transaction (as it does not distinguish between them). The user must re-add other records that should have not been deleted (if desired).

The IDT layout must be specified in the UST-Definition Section of the SDF file and not the Files- Definition. The latter is used for unsynchronized flat-file input. Refer to the *User Source Table* section of the *DESIGN GUIDE*.

NSA Transaction Table

NSA is an acronym for No Source Access. The NSA Transaction Table (NSATT) is used to store transactions pertaining to a source database to which we have no source access (non-SQL or unsupported DBMS).

The transaction data contained within the NSATT is similar to a Flat-File. It contains the following columns:

Column	Max Length	Description
SYSTEM	32	System Name
SEQ	32	Transaction Sequence
OP	1	Operation Code ('A' or 'D')
IDT_NAME	32	IDT Name
IDT_REC	DBMS	dependent IDT Record

The SystemName, Operation Code, IDT Name, and IDT Record are identical to those already described in the Flat-File Layout.

Transaction Sequence uses different semantics. In the `NSATT` it is not a number but an alphanumeric string. SEQ is implemented as a `VARCHAR` column and therefore its ordering is determined by the collation order of the DBMS.

It is critical that the sequence number of a newly added row is greater than the sequence number of all rows already in the table. Failing to do so may lead to incorrect synchronization results.

Note: The `syncstop` script cannot be used to stop a Synchronizer that is processing the NSA TT. The only supported mechanism is the **Stop** button on the Console.

Integrity Checking

The Update Synchronizer can be used to check the integrity of the IDT. When started with the `-i` switch, the Update Synchronizer compares the current contents of the User Source Tables against the current state of the IDT and report any differences. It can also check the integrity of the IDT vs IDXs.

This process does not take into account the following anomalies that might cause an incorrect error report:

- unapplied transactions held in the Transaction Table
- updates to the UST that have occurred while opening the cursors that reads it.

Therefore any errors reported may be transient. The best way to check is to run the Update Synchronizer to process all known updates and run the integrity checker a second time to see if the errors are transient.

Note: The IDT vs IDX integrity check confirms that every IDT row has at least one IDX entry. If the IDX has been built with the NO-NULL-KEY option, some IDT rows may not have a corresponding row in the IDX as they generated NULL keys. The integrity checker flags this case as an error when in reality no error exists.

Syntax

The integrity checker is invoked with the `-i` switch:

```
-i[IDT[,IDX]]
```

If `-i` is specified without an IDT nor IDX name, it will check all IDTs within the System, and all IDXs against each IDT.

If an IDT name is provided, that specific IDT will be checked against all IDXs.

If an IDT and IDX name is provided, only that specific IDT/IDX combination will be checked.

The optional `-k` switch can be used to display detailed (field level information) for any erroneous records.

Performance

Update Synchronization is inherently expensive because IIR denormalizes the USTs in order to provide very fast search performance. The disadvantage is that updates are slower. Conversely, had IIR not denormalized the data, the searches would be much slower and updates would be faster. This is a tradeoff in the design.

The following sections describe methods to optimize update performance.

Overlap Processing

Run one Synchronizer per IDT so that update processing is overlapped for multiple IDTs in a System (`-e` switch).

IDT/IDX Design

The design of each IDT directly impacts the Synchronizer's performance. It can be improved by minimizing the use of expensive features where possible:

- reduce the amount of denormalization (the number of joins and especially the number of one:many joins).
- use Flattening where possible.
- avoid non-unique PKs
- reduce the size of the IDT and IDX records by ensuring that only those columns required for key generation and matching are sourced from USTs.

- reduce the number of keys per IDT record (standard or limited key options)
- do not use the Auto-Id feature for synchronized tables

Compressed Key Data

Specify a Compress-Key-Data value for each IDX that minimizes the number of database blocks required to store it. This will improve the performance of searching and updating the IDXs. A poorly selected value for Compress-Key-Data can easily double or triple the amount of I/O required.

Use the histkg utility to analyze the report file created by the Table Loader. For each IDX, determine the `segLen` value that minimizes `DB-blocks`. This usually occurs when `segs/key` is a multiple of a whole number. The best value occurs when `segs/key` is near 1.0.

Refer to the *Compressed Key Data* section in the *DESIGN GUIDE* for details.

Network Issues

Reduce network overhead by running the Synchronizer and the Rulebase Server on the same machine as the database server. If this is not possible, tune your network parameters to optimize throughput.

SQL*Net / Net8

The SDU parameter controls the network packet size. The default value of 2048 is slightly too small to hold a complete Transaction-Table record (2178 bytes). This causes packet fragmentation as the Server must send two packets to the Client (Update Synchronizer) to return each transaction.

Increase the SDU to at least 2200. A value of 3000 is recommended for Ethernet networks, as it is a multiple of Ethernet frame size (1500 bytes).

Change `$ORACLE_HOME/network/admin/listener.ora` (on the server) to include the SDU parameter and stop/start the listener using the lsnrctl utility. For example:

```
SID_LIST_LISTENER =
(SID_LIST =
    (SID_DESC =
            (SDU=3000)
            (GLOBAL_DBNAME= ssa16.)
            (ORACLE_HOME= /home/oracle/u01/app/oracle/product/8.0.5)
            (SID_NAME = dba)
```

You must also change the client side configuration file to specify a matching SDU, as the SDU is negotiated down to the smallest value when the client connects to the server. Change `$ORACLE_HOME/network/admin/tnsnames.ora` to add the SDU parm. For example:

```
ssa16 =
(DESCRIPTION =
        (SDU=3000)
        (ADDRESS = (PROTOCOL= TCP)(Host= ssa16)(Port= 1521))
        (CONNECT_DATA = (SID = dba))
)
```

However on fast (low traffic) networks, this will only provide a minor performance boost.

A major improvement comes from specifying the TCP parameter NoDelay (assuming that TCP/IP is the protocol being used). This tells TCP to flush buffers without waiting for them to fill. Modify (or create) `$ORACLE_HOME/network/admin/protocol.ora` and add a line to it that specifies, `tcp.nodelay=yes`

Optimizer Statistics

Ensure that DBMS optimizer statistics are up-to-date. This is especially important if a batch job has added many new transactions to the Synchronizer's Transaction Table. Use the SQL ANALYZE command to update the Optimizer's statistics, or use the Synchronizer's `-o` switch to regularly update statistics automatically.

```
analyze table ids_upd_sync_txn estimate statistics
```

The USTs and IDT/IDXs should also be analyzed regularly. The Table loader will automatically analyze the IDTs and IDXs after they have been loaded.

Commit Rate

An appropriate commit rate needs to be selected by tuning.

In general, a high commit rate will provide better transaction throughput. However, too high a rate may cause the database to run out of rollback space in a multi-user update environment, and updated records won't be visible to searches for long periods. A database failure that interrupts Synchronization processing will mean more work will be repeated when the Synchronizer is restarted.

A very low commit rate will cause frequent database I/O that slows down the Synchronization process.

The commit rate must tend toward a low value when multiple Synchronizers are running simultaneously against the same database. High commit rates will create contention for the table that allocates the unique record numbers for each IDT, causing lots of database I/O to maintain "read consistency".

Flat-File Synchronization

The two-phase commit table (`IDS_2PC`) is used to record the file-names of the files that have been applied. File-names are not removed from this table, so that if an input-file is accidentally reused, the situation will be recognized and the transactions will be ignored.

The consequence of this is that the table will grow at the rate of one row per input-file processed. The table is not normally indexed in order to optimize update performance when very few rows are present (as is the case for user-source synchronization). As the number of rows grows, performance will slowly degrade. To avoid this problem, create an index on the table:

```
CREATE INDEX IDS_2PC_I ON IDS_2PC (ID);
ANALYZE TABLE IDS_2PC ESTIMATE STATISTICS;
```

Timing Window

When the IIR Table Loader creates an IDT it creates triggers on the User Source Tables, commits them and opens a cursor to extract data from the USTs. A very small timing window exists between the commit and the opening of the cursor.

If a user transaction starts, adds a new record and commits inside this window, the trigger is fired and an "add" transaction is logged to the transaction table. The cursor used to unload the UST records will also see this new record so it is added to the IDT as part of the initial load process. When the Synchronizer starts processing transactions and attempts to "add" the same record it will detect that the record already exists and will terminate with a PK violation error.

In the unlikely event that this happens, you may delete the transaction using the steps outlined in the *Repairing a UST* section.

CHAPTER 7

Globalization

This chapter includes the following topics:

- Overview, 102
- Character Sets, 102
- Database Support for UNICODE, 103
- Binary Mode Utilities, 105
- Loading IDTs, 106
- IIR Clients, 107
- Debugging a Search, 108
- Miscellaneous Tips, 109

Overview

This chapter deals with IIR issues relating to multiple languages, character sets and `UNICODE`.

Each DBMS that IIR supports handles those issues differently. This chapter discusses the issues in a general way first and then presents DBMS specific issues.

Character Sets

A `character set` is used to represent all characters (or code points) in a language or script. The first character sets were single byte, meaning that they could only define a maximum of 256 characters.

A `code point` is simply a binary value that represents a character in a character set. ASCII and EBCDIC are examples of two single byte character sets that use different code points to represent the same set of characters. For example, the code-point 0x41 represents the ASCII letter 'A' but in EBCDIC, the same letter is represented by 0xC1.

Some complex scripts contain more than 256 characters, so they need to use multiple bytes to represent a single character. The most common multi-byte character set is UNICODE.

The characters in a character set may be encoded in many ways. For example, a single byte character set could use a 7-bit or 8-bit encoding. A multi-byte character set could use a fixed width, variable width, or shift-sensitive variable-width encoding.

UNICODE Encoding

UNICODE supports three main encodings:

`UCS-2` a 2 byte fixed width encoding.

`UTF-16`a 2 byte fixed width encoding. In order to increase the range of characters that can be represented, a character may be followed by a supplemental character increasing the length to 4 bytes.

`UTF-8` a variable length encoding ranging from 1 to 4 bytes in length. 7-bit ASCII characters are represented by a single byte in UTF-8 and use the same code-points. Therefore ASCII characters are indistinguishable from their UTF-8 encoded, Unicode counterparts.

Operating System Character Set

The operating system must have the appropriate character sets installed to be able to render the characters properly. Install a native language version of the operating system, or on Win32 install the English version with additional character sets.

Microsoft Windows

On Windows operating systems your Locale determines the ANSI character set used for rendering text in GUI applications. The corresponding OEM character set is used by console applications (those that run in a DOS Box). For example, U.S. English uses ANSI code page 1252 and OEM code page 437.

The Locale also determines the way numbers, currency, time and dates are displayed. The Locale is set using the **Regional Options/Setting** dialog, which is accessible from the Control Panel.

Note: The Input Locale (as distinct from the Locale) determines your keyboard to character setting mapping.

MS-DOS Box

In order to render characters using different Locales from within an MS-DOS Box, select a True Type font. Raster Fonts cannot be used.

OEM code pages can be set explicitly with the `chcp` utility from within a DOS Box. For example:

```
C:\>chcp /?
Displays or sets the active code page number.

CHCP [nnn]

        nnn Specifies a code page number.

Type CHCP without a parameter to display the active code page number.

C:>chcp
Active code page: 437
```

Rendering CJK with English Locales

A useful tool for displaying CJK characters on an English/Western version of Windows is NJWIN's CJK Viewer.

Database Support for UNICODE

There are two main ways that databases support UNICODE characters:

`Database Level` Some databases store all columns of all tables as UNICODE. This allows multiple database clients to use different character sets and have their data stored without loss since UNICODE is a superset of all client character sets.

`Column Level` Some databases allow individual columns in a table to be defined as UNICODE, while others are not. The UNICODE data types are usually preceded by the letter 'N' (for National). For example NCHAR, NVARCHAR, NCLOB, etc.

Oracle

Oracle Database

UNICODE support for Oracle databases may be implemented in two ways by defining:

- the database character set as UTF8 so that UTF-8 encoded characters may be stored in all CHAR data types (CHAR, VARCHAR2, CLOB), or
- individual columns as UNICODE data types (NCHAR, NVARCHAR2, NCLOB). This allows you to add UNICODE support incrementally for only some specific columns in your tables.

Oracle databases define two character sets when the database is created:

- database character set (`NLS_CHARACTERSET`), and
- the character set used for NCHAR or NVARCHAR columns (`NLS_NCHAR_CHARACTERSET`). Valid values are UTF8 or AL16UTF16.

The following SQL*Plus script can be used to determine how the database was configured:

```
select parameter, substr(value,1,20) from NLS_DATABASE_PARAMETERS;

PARAMETER                      SUBSTR(VALUE,1,20)
------------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY
NLS_CURRENCY                   $
NLS_ISO_CURRENCY
NLS_NUMERIC_CHARACTERS         .,
NLS_CHARACTERSET               UTF8
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-RR
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT BINARY
NLS_TIME_FORMAT                HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT           DD-MON-RR HH.MI.SSXF
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZH:T
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXF
NLS_DUAL_CURRENCY              $
NLS_COMP                       BINARY
NLS_NCHAR_CHARACTERSET         UTF8
NLS_RDBMS_VERSION              8.1.7.0.0

18 rows selected.
```

Oracle Client

Although Oracle can store data as UNICODE characters, the client application may not be aware of this because data are converted upon retrieval. The environment variable *NLS_LANG* defines the character set of the database client. This character set is not necessarily UNICODE, although UNICODE is a valid option. `NLS_LANG` has the format `X_Y.Z` where

`X` is the value of `NLS_LANGUAGE`

`Y` is the value of `NLS_TERRITORY`, and

`Z` is the value of `NLS_CHARACTERSET`

Multi-byte data in a non-UNICODE column

It is possible to store multi-byte characters in a non-UNICODE database and/or column. Data stored in `CHAR/VARCHAR` columns is normally translated between the client and server's character sets when transferred between client and server. But if the client and database are configured to use the same character set, no conversion is performed. This makes it possible to store multi-byte characters within `CHAR/VARCHAR` columns without interference from the DBMS.

Microsoft SQL Server

A column defined as a non-UNICODE data type can only store a single code page (character set). The code page is determined by the collation of the column defined at table creation time, or if none was specified, the collation of the database.

Columns defined using UNICODE data types such as `NCHAR` and `NVARCHAR` can store/retrieve UNICODE characters. They always use an `UCS-2/UTF-16` encoding. MSQ database clients work directly with "raw" UNICODE characters, without translation to a client character set.

UDB

For UDB, the database must be created as a UNICODE database. By using code set utf-8, Unicode data will be stored in UTF-8 form.

The easiest way to check that this is the case is with the following command:

```
db2 get database config for mydb
```

The response will be something like:

```
                        Database Configuration for Database mydb

 Database configuration release level     = 0x0a00
 Database release level                                = 0x0a00
 Database territory                                      = AU
 Database code page                                      = 1208
 Database code set                                       = UTF-8
 Database country/region code                    = 61
```

The data file used by the load process will be in UTF-16 which will be converted to UTF-8 by UDB.

Binary Mode Utilities

Multi-byte data should be treated as binary data. That is, DBMS and IIR utilities must be informed that they are operating on binary data so that they read the data using binary mode file I/O.

By default, IIR and Oracle utilities assume they are operating on text data, and will read and write files in text mode. If a character in the input file matches a newline (CR/LF) or End-of-File marker (Ctrl-Z), the input file will not be read correctly and records may be accidentally split and/or the whole input file may not be read. This is more likely to occur with `UTF-16/UCS-2` data.

Loading IDTs

The IIR Table Loader generates DBMS load files in delimited text format by default. Specify the Loader-Definition option `FIXED` to generate fixed length binary files instead.

To verify that the input data has been read and processed properly, you may specify the Loader- Definition option `Keep-Temp`. This will prevent the DBMS loader files from being deleted after the load completes so that they may be examined.

Flat-File Input

If input data is read from a flat-file, make sure that

- the file contains fixed length records
- `FORMAT=Binary` is specified in the `Logical-File-Definition`
- the format of the input records matches the Input-View
- the record length is the sum of the field lengths in the View

MSQ

Data loaded from a flat-file into `CHAR` or `VARCHAR` columns will be translated from the client's code page into UNICODE, transferred to the server and then translated into the server's code page and stored.

The client's code page should be identical to the server's code page, otherwise the conversion could be lossy.

If the character set of the data file does not match the client's code page (Locale) specify the DATABASEOPTIONS= IDTCP parameter to specify the code page of the data.

User Source Table Input

During data extraction from a User Source Table's `CHAR/VARCHAR` columns, the data is translated from the Server's code page to the client's code page (Table Loader). The client's code page should be identical to the server's code page; otherwise the conversion could be lossy.

Oracle

A safe approach is to use a UNICODE character set for the database and to specify a UNICODE character set for the client (IIR Servers and utilities). IIR automatically requests UNICODE data to be returned as UTF-16 .This ensures that no lossy conversions are performed when reading/writing data to/from the database.

The mass loader file generated by the Table Loader will automatically use a `Fixed` length format when UNICODE columns are present.

MSQ

`NCHAR` and `NVARCHAR` columns are not converted. They remain in UNICODE format.

Target Column Size

The format and length of an IDT column defaults to the same values as the source column. In most cases, this is adequate. However, if the source and target databases do not use the same character set, it may be necessary to increase the size of the target column to accommodate the change of encoding.

For example, suppose a source column is defined as CHAR(5) encoded in a Central European character set such as Windows Code Page 1250. Suppose a particular row in the source table contains 5 bytes of data, with four of them being Latin characters (hexadecimal values <0x7F) and one of the characters being the Latin character A with an Acute (= 0xC1 = U+00C1).

When encoded in this character set, the data only requires 5 bytes. However, if it is now stored in an IDT on a database that uses UTF-8 as its character set, the data will be converted. The Latin characters will still only

require one byte when expressed as UTF-8 but the `A + Acute` will be encoded using 2 bytes, with the total storage requirement being 6 bytes.

If the default column size is insufficient, use a length override in the definition of the target column to increase its size.

IIR Clients

Custom Search Client

The search API **ids_set_encoding** is used to inform the Search Server of the encoding used by the client application for UNICODE columns (data type 'W').

If an encoding has been specified that is different to the encoding used by the Search Server (UTF-16), the search data will be converted prior to searching and similarly, prior to the return of the result set.

UTF-16 UTF-8 conversions occur on the machine running the Search Server. UTF-16 data is assumed to be encoded in the byte order of the Search Server's machine.

Note: If the search client requests UTF-16 data (the default for 'W' columns), they will be encoded using the native byte order of the Search Server, which may be different to the byte order of the client machine.

Oracle

Oracle W fields are stored as UNICODE in the database's national character set using `NCHAR/NVARCHAR2` data types. Upon retrieval by the Search Server the data is converted to UTF-16 (if necessary). If the caller's search data is encoded differently, the caller must call **ids_set_encoding** to inform the Search Server.

MSQ

IIR stores and retrieves data for W fields as UNICODE characters encoded as UTF-16. If the caller's search data is encoded differently, the caller must call **ids_set_encoding** to inform the Search Server.

For example, if the client's search data is encoded as UTF-8 the Search Server will convert incoming data from UTF-8 to UTF-16, perform a search and translate the search results back to UTF-8.

Relate

The batch search client `relate` will need to read the input file in binary mode (`-b` switch). Make sure that the input file contains fixed length records matching the input view length (`-i` switch).

The output is written in binary mode when `-b` is specified with fixed length records of IDT record length, plus any header information. The `-o` switch determines the exact layout.

You may wish to add newlines to the output by specifying `-t`. This will make the output easier to view in a text editor, but it is only useful if the binary data does not contain any newline characters; otherwise output lines will be split.

The search trace facility is enabled with the `--3` switch. It automatically detects non-printable characters when processing the Search and File records and will log them in hexadecimal format when necessary.

Search records should use UTF-16 for W columns. If they use UTF-8, specify the `-e` switch to inform the Search Server of the encoding used for the input fields.

Java Search Client

Java applications use UTF-8 encoding for UNICODE data. The IIR GUI Search Client automatically informs the Search Server that incoming data is encoded as UTF-8.

Synchronizer

The Update Synchronizer can handle binary data in `CHAR/VARCHAR2` columns with one `exception: columns` defined as Primary Keys (PK) must not contain the NUL (binary zero, 0x00) character.

Columns of type W cannot be used as PK fields.

SSA-NAME3 V2

When indexing or searching a field containing Unicode, specify Key-Logic and Search-Logic Controls to inform SSA-NAME3 that UTF-16 data is present:

```
Controls ("UNICODE_ENCODING=6")
```

Debugging a Search

UNICODE data is notoriously difficult to handle. It requires an intimate knowledge of the data and micro-management of the search process to avoid inappropriate conversions and to request conversions when necessary.

Batch Searches

The most reliable approach to debugging a new search is to use a batch search client such as relate. A batch client has several advantages over an online search client:

- the input file can be viewed and/or manipulated with a hex editor, so you have precise control over the input data.
- search records read from a file are not subject to conversions performed by the Operating System. Use fixed length records and specify `-b` to read the file in binary mode. This avoids characters being interpreted as `CR/LF` and being converted to `LF`.
- only server side tracing is necessary to verify the search process.

Online Searches

In contrast, an online search client has less control over the input data because the Operating System may perform unexpected conversions while the data is entered:

- If the data is typed, the characters that end up in the input buffer are dependent on the keyboard driver, language, and locale being set correctly.
- A cut and paste operation may also perform conversions on the data. It may be corrupted even before you hit the enter key to start the search.
- The correct search results may look wrong if the locale has been configured incorrectly due to incorrect rendering of the characters.
- If you do decide to use the online client, make sure to use the IIR 950 client and enable the client side logging facility to produce a hex dump of the input data.

Server-side Search Tracing

The *Search Performance* chapter of the *DESIGN GUIDE* documents provide informatation on how to enable the Search Trace facility. This feature is particularly useful when debugging `UNICODE` problems as it logs the Search Record and File Records in hexadecimal format. It also displays the records before and after view conversion and/or `UNICODE` encoding conversion.

Miscellaneous Tips

Loading User Source Tables - Oracle

When loading data to User Source Tables, make sure that the input file contains fixed length records and instruct the DBMS loader to read the file in binary mode.

For Oracle this is done with the FIX option. For example, if the input file contains fixed length records 16 bytes long encoded as UTF-8, the following control file could be used:

```
LOAD DATA
CHARACTERSET AL32UTF8
LENGTH SEMANTICS BYTE
INFILE 'testx182.ut8' "FIX 16"
REPLACE
INTO TABLE TESTX182A
(
                        NAME         POSITION( 1: 16) CHAR
)
```

If the input file contains fixed length records 16 bytes long encoded as UTF-16, the following control file could be used.

Note: The byte order must be specified so that SQL*Loader can convert it to match the DBMS Server machine's byte order, if necessary.

```
LOAD DATA
CHARACTERSET UTF16
LENGTH SEMANTICS BYTE
BYTEORDER LITTLE
INFILE 'testx182.u6l' "FIX 16"
REPLACE
INTO TABLE TESTX182B
(
                        NAME         POSITION( 1: 16) CHAR
)
```

Validating Loaded Data

Binary data stored in `CHAR` or `VARCHAR2` columns may be displayed using a number of methods:

It is easiest to work in hexadecimal format by converting the `CHAR` data to RAW, as SQL*Plus automatically displays RAW data in hexadecimal format. A sample package called **ids_conv** is provided that will convert CHAR to RAW and vice versa. For example, the following script installs the package and calls it:

```
@%SSABIN%\idsconv9.sql
select row_id, ids_conv.chartoraw(name) from T106;
```

It produces output similar to this:

```
ROW_ID IDS_CONV.CHARTORAW(NAME)
------ --------------------------------------------------------------------------
```

```
                    1
28C3C0B9FA29CFD6B4FAB6ABCEF7B7BDB9ABCBBE2020202020202020202020202020202020202
                    2
28C3C0B9FA29CFD6B4FAB6ABCEF7D1D4B9ABCBBE2020202020202020202020202020202020202
                    3
B0AED2C0CBB9D6D0B9FAB7A2B5E7D3D0CFDEB9ABCBBE2020202020202020202020202020202020
                    4
B0AED2C0CBB9D6D0B9FAD3D0CFDEB9ABCBBE2020202020202020202020202020202020202020202
                    5
B0D7B5C3B2B9B1A6B8D6D3D0CFDEB9ABCBBEC9CFBAA3C1AAC2E7B4A620202020202020202020202
                    6
B0B2B4EFD0C528C9CFBAA329C6F3D2B5D7C9D1AFD3D0CFDEB9ABCBBE20202020202020202020202
                    7
B0B2B4EFD0C52EBBAAC7BFBBE1BCC6CAA6CAC2CEF1CBF9202020202020202020202020202020202
                    8
B0B2B4EFD0C5BCC6CBE3BBFAD0C5CFA2CFB5CDB3B1B1BEA9D3D0CFDEB9ABCBBE202020202020202
                    9
B0B2B4EFD0C5C7E5BBAAB4F3D1A7D0C5CFA2CFB5CDB328B1B1BEA929D3D0CFDEB9ABCBBE2020202
                   10
B0B2B5C2C0FBD3D0CFDEB9ABCBBE2020202020202020202020202020202020202020202020202020202
```

An alternative is to use the `DUMP` function to display table data in hexadecimal format. This also displays the name of the database character set used to store the column. For example,

```
select dump(ids_name,1016) from idt182;

DUMP(IDS_NAME,1016)
--------------------------------------------------
Typ=1 Len=6 CharacterSet=AL16UTF16: 30,a0,30,a1,30,a2
Typ=1 Len=6 CharacterSet=AL16UTF16: 30,f0,30,f1,30,f2
```

dd

The dd utility can be used to add newlines to a file containing fixed length records. Use

```
dd InputFile OutputFile 0 RecLen -a -b
```

CHAPTER 8

Siebel Connector

This chapter includes the following topics:

- Overview, 111
- Configuring Siebel, 111
- Configuring IIR, 118

Overview

You can configure Identity Resolution to load, synchronize, and search against data stored in a Siebel 7.7 CRM application. This chapter provides detailed instructions for integrating Identity Resolution with Siebel.

Siebel application data is held in an Oracle, UDB or Microsoft SQL Server database. At a physical level, the data are held in base tables. However, base tables are not accessed directly. Instead, Siebel provides a higher level of data abstraction with its Object Manager (OM). The joins base tables to provide highlevel Integration Objects (IO) that the user works with.

As Siebel prohibits the creation of triggers on base tables, Identity Resolution treats Siebel as a No Source Access (NSA) style of database. The fundamental unit of data that can be extracted or synchronized is an IO, which is mapped 1:1 to an Identity Resolution IDT.

To extract and synchronize data, the Siebel administrator must first define an IO using Siebel Tools. A matching IDT-Definition is created in an Identity Resolution System.

Identity Resolution provides a Siebel Workflow to extract data using the Object Manager. The workflow will query Siebel to extract the IO data, encode them using XML and write them to a flat-file. This file can then be loaded into Identity Resolution.

Synchronization workflows (activated by Run-Time Events) are provided to pass synchronization messages to Identity Resolution whenever an object is added, deleted, or modified. The messages are encoded as XML and sent over a socket using HTTP to the Identity Resolution XS Server. The XS Server stores transactions in the NSA Transaction Table for processing by the Update Synchronizer.

Configuring Siebel

The Siebel application must be configured using Siebel Tools. The following section describes the process.

Constructing Load Data

To produce an XML file for the load process, the Workflow `Launch Build Load File` will need to be invoked. The following steps outline what is required to invoke the workflow.

- Create and Compile an appropriate integration Object. See the *Integration Object* section.
- Create an appropriate IIR System to match your Integration Object. See the *Configuring IIR* section.
- Import and Compile The IIR Business Service.
- Import, Deploy and Activate the IIR Workflows. See the *Workflows* section.
- Create an Action set for the Workflow. See *Load Action Set* section.
- Create a Runtime Event associated with the Action Set. It is left to the user to decide what type event to use. The user may for instance decide to create a **MiniButton** within an applet and invoke the Workflow based on this button.

Synchronization Setup

Several Workflow processes are provided to synchronize changes to the BC with the IIR IDT. These Workflows must be invoked with the appropriate BC Events. The following steps outline what is required to set up this synchronization process.

- Create and Compile an appropriate Integration Object. See the *Integration Object* section.
- Create an appropriate IIR System to match your Integration Object. See the *Configuring IIR* section.
- Import and Compile the IIR Business Service.
- Import, Deploy and Activate the IIR Workflows. See the *Workflows* section.
- Create Action Sets for the Workflows. See the *Synchronization Action Sets* section.
- Create appropriate Run Time Events which use the Action Sets. See the *Synchronization Run Time Events*section.
- Reload Run Time Events.

Integration Object

The basic mapping of data contained in a Siebel Business Component (BC) to an IDT is through an Integration Object (IO). A Siebel Integration Object will be set up with all the fields in the Business Component that are desired in the IDT. Then the IDT can be set up to match the XML Tag names for the Integration Object. What follows is an example of an XML message based on such an Integration Object.

The corresponding IIR UST definition can be found in the *Configuring IIR* section. You must always include `RowId` as an active field in your IO as it will be used as the primary key in the IDT.

```
<?xml version="1.0" encoding="UTF-16"?>
<?Siebel-Property-Set EscapeNames="false"?>
<SiebelMessage MessageId="1-3HPY" IDS_OP="A" IDS_SYSTEM="testx218
        IDS_IDT="IDS_01_CONTACT" MessageType="Integration Object"
IntObjectName="ISS IO Contact"
IntObjectFormat="Siebel Hierarchical">
                <ListOfIssIoContact>
                    <Contact>
                        <Alias></Alias>
                        <BirthDate>01/14/1932 00:00:00</BirthDate>
                        <FirstName>Jean</FirstName>
                        <LastName>Murasawa</LastName>
                        <MiddleName></MiddleName>
                        <City></City>
                        <Country></Country>
```

```
                        <PostalCode>765048832</PostalCode>
                        <StreetAddress></StreetAddress>
                        <RowId>12-ZT80P</RowId>
                    </Contact>
                </ListOfIssIoContact>
</SiebelMessage>
```

IIR Business Service

All the IIR Workflows require the IIR Business Service (IIR Utility Service). You will need to import and compile this in order to use the IIR Workflows. This has to be done prior to importing the Workflow Processes. See the *Workflows* section. It can be found in the `issutilityservice.sif` archive file in the `siebel/busservs` directory of your IIR installation.

Error Handling

The Workflow `ISSErrorHandler` is used by all the IIR Workflows to log errors to a file. The default for this log file is `/tmp/isserror.log`. The file name may be changed by modifying the Workflow. Simply change the File Name value for input in the `Write to Error Log` step.

Workflows

The IIR Workflows can be found in the siebel/workflws directory of the IIR installation. You will need to import all 9 Workflows into Tools.

Note: You must import the workflows in order of their dependencies.

After importing, you may need to modify the **ISSErrorHandler** Workflow (See the *Error Handling* section). Then click **Deploy** for each of the workflows. You will then activate them from the client by navigating to the **Repository Workflow Processes** Screen.

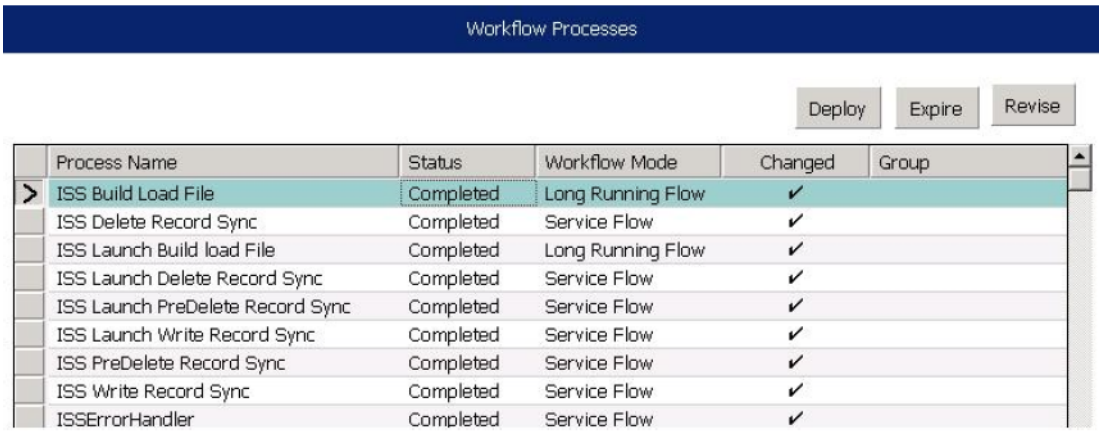
Once you have click **Activate** on all the Workflows they can be found in the Active Workflow Processes list. More detailed instructions on importing and activating workflows can be found in Siebel's Bookshelf. The following is the list of all the IIR workflows.

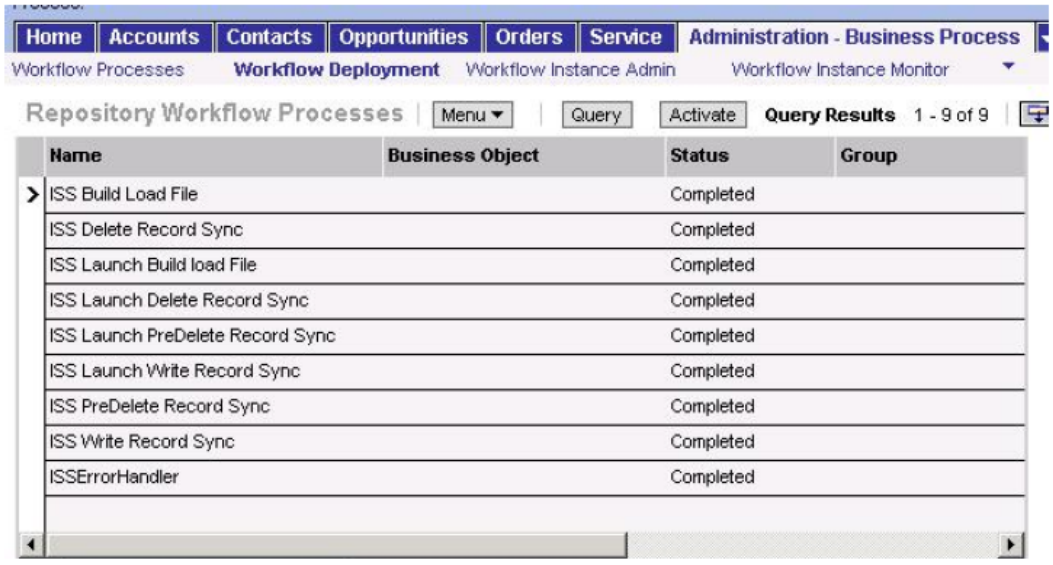
```
IIR Build Load File
IIR Delete Record Sync
IIR Launch Build Load File
IIR Launch Delete Record Sync
IIR Launch PreDelete Record Sync
IIR Launch Write Record Sync
IIR PreDelete Record Sync
IIR Write Record Sync
IIRErrorHandler
```

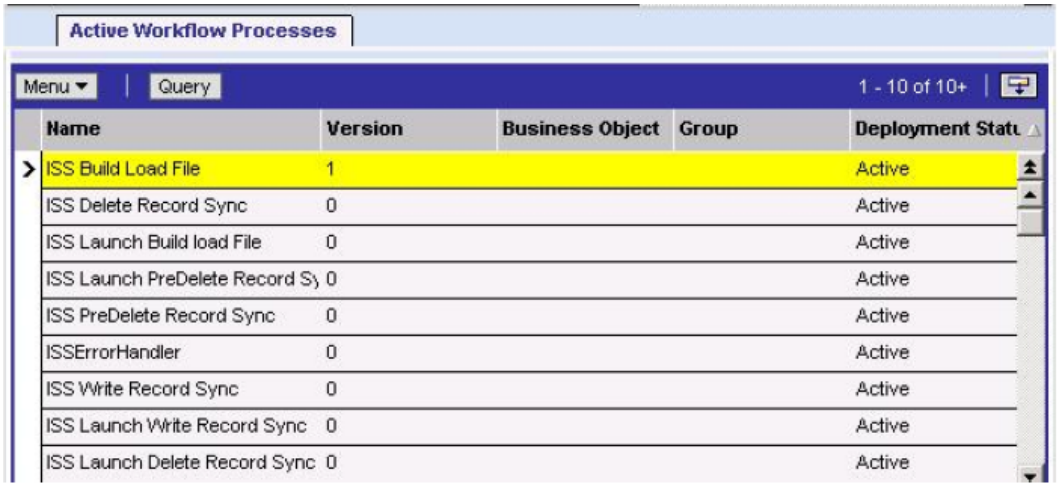
There are dependencies between these Workflows. Siebel will issue validation warning and errors when deploying a Workflow if any required Workflows are not already deployed or were not imported prior to importing the current Workflow. You must import and deploy all required workflows first.

The dependancy of the Workflows are:

- `IIRErrorHandler` is required by all other Workflows.
- `IIR Launch Build Load File` requires `IIR Build Load File`.
- `IIR Launch Delete Record Sync` requires `IIR Delete Record Sync`.
- `IIR Launch PreDelete Record Sync` requires `IIR PreDelete Record Sync`.
- `IIR Launch Write Record Sync` requires `IIR Write Record Sync`.


Workflow Processes
Deploy
Expire
Revise
Process Name
Status
Workflow Mode
Changed
Group
ISS Build Load File
Completed
Long Running Flow
ISS Delete Record Sync
Completed
Service Flow
ISS Launch Build load File
Completed
Long Running Flow
ISS Launch Delete Record Sync
Completed
Service Flow
ISS Launch PreDelete Record Sync
Completed
Service Flow
ISS Launch Write Record Sync
Completed
Service Flow
ISS PreDelete Record Sync
Completed
Service Flow
ISS Write Record Sync
Completed
Service Flow
ISSErrorHandler
Completed
Service Flow

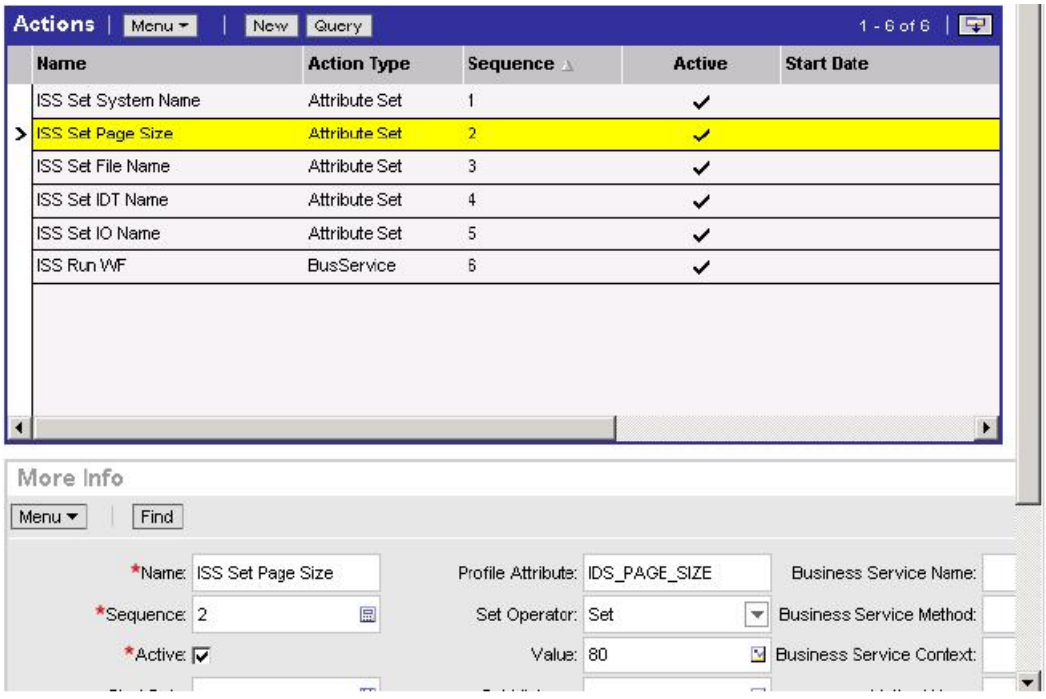

Home
Accounts
Contacts
Opportunities
Orders
Service
Administration - Business Process
Workflow Processes
Workflow Deployment
Workflow Instance Admin
Workflow Instance Monitor
Repository Workflow Processes
Menu
Query
Activate
Query Results
1 - 9 of 9
Name
Business Object
Status
Group
ISS Build Load File
Completed
ISS Delete Record Sync
Completed
ISS Launch Build load File
Completed
ISS Launch Delete Record Sync
Completed
ISS Launch PreDelete Record Sync
Completed
ISS Launch Write Record Sync
Completed
ISS PreDelete Record Sync
Completed
ISS Write Record Sync
Completed
ISSErrorHandler
Completed


Active Workflow Processes
Menu
Query
1 - 10 of 10+
Name
Version
Business Object
Group
Deployment Statu
ISS Build Load File
1
Active
ISS Delete Record Sync
0
Active
ISS Launch Build load File
0
Active
ISS Launch PreDelete Record S
0
Active
ISS PreDelete Record Sync
0
Active
ISSErrorHandler
0
Active
ISS Write Record Sync
0
Active
ISS Launch Write Record Sync
0
Active
ISS Launch Delete Record Sync
0
Active

Load Action Set

You Must Create an Actions Set for Calling the Workflow: `IIR Launch Build Load File`. This Workflow requires some profile attributes to be set (see *Profile Attributes*section). Appropriate Actions must be added to set these Profile Attributes.

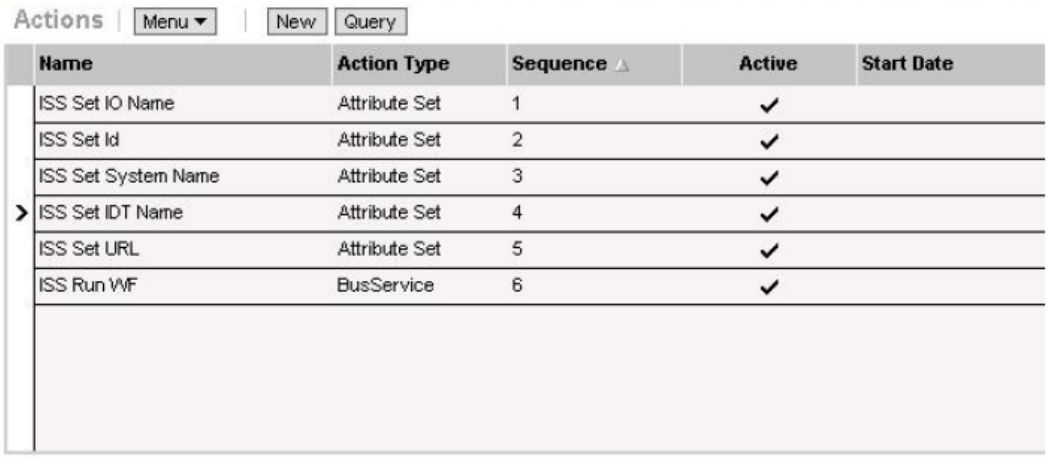
You must ensure that the action that triggers the Workflow Process is last in sequence. We recommend naming this Action Sets with the prefix `ISSLOAD` then add the name of the Business Component you are working with (example, `ISSLOAD Contact`). This Action set will then be associated with an appropriate Runtime event (Example: The click of a Mini-Button).



Synchronization Action Sets

You must create Actions Sets for calling the Synchronization Workflows. You will need three Action Sets. One for the Pre Delete Event which calls the Workflow `IIR Launch PreDelete Record Sync`, one for the Delete Event that calls the Workflow `IIR Launch Delete Record Sync`, and one more for the Write record event which calls the Workflow `IIR Launch Write Record Sync`. These Workflows require some profile attributes to be set (See the *Profile Attributes* section). Appropriate actions must be added to set these Profile Attributes.

You must ensure that the action that triggers the Workflow Process is last in sequence. We recommend naming these Action Sets with the prefix `ISSSYNC` then with the event type they will be associated with and then finally add the name of the Business Component you are working with. For example, `ISSSYNC WriteRecord Contact`.



Synchronization Run Time Events

You need to create runtime events that use the Action Sets you have created. These will all be of the Type `BusComp`. The Object Name will be set to name of the Business Component you are working with. The Events will be set to `PreDeleteRecord`, `DeleteRecord` and `WriteRecord`. After you have created these events you will need to reload the Run Time Events.

Profile Attributes

The following tables show the profile attributes used by the IIR Workflows.

Attribute Name	Description
IDS_SYSTEM	The name of the corresponding IIR system
IDS_IDT	The fully-decorated name of the corresponding IDT database table. Example: **IDS_01_CONTACT**
IDS_IO_NAME	The name of the Integration Object to be used
IDS_IO_ID	The Id of the primary business Component for the Integration Object, that is `[Id]`
IDS_URL	The URL of the XS Server
IDS_PAGE_SIZE	The page file size used by the EAI Siebel Adapter Business Service
IDS_LOADFILE	The full path of the XML load file to create

Attribute Name	Launch PreDelete Record Sync	Launch Delete Record Sync	Launch Write Record Sync	Launch Build Load File
IDS_SYSTEM	Required		Required	Required
IDS_IDT	Required		Required	Required
IDS_IO_NAME	Required		Required	Required
IDS_IO_ID	Required		Required	
IDS_URL		Required	Required	
IDS_PAGE_SIZE				Optional
IDS_LOADFILE				Required

Configuring IIR

Having defined IOs using Siebel Tools, we must now create an IIR System containing equivalent IDTs.

System Definition

Data will be loaded from a Flat-File containing XML messages. Define an IDT in the User-Source-Tables section of the SDF for each IO.

- All field names must correspond to the names of the fields in the IO. Any fields present in the IO but not listed in the IDT definition will be ignored.
- Field types must be W (wide format) as the XML messages contain Unicode. Field lengths are specified in bytes, not Unicode characters.
- If synchronization is required, the `sync_clause` should specify `SYNC REPLACE_DUPLICATE_PK TXN-SOURCE NSA`, otherwise `NOSYNC`.
- Specify the Siebel `RowId` as the primary key (`PK`).
- The Logical-File-Definition describing the flat file must specify `FORMAT=XML` and `VIEW=<IDTName>1`(the IDT name suffixed by "1" which is the automatically generated view name).
- The Loader-Definition must specify `OPTIONS=FIXED`.
- The Controls parameter of the `IDX-Definition` and `Search-Definition` associated with the IDT must specify `UNICODE-ENCODING=6`(for MSQ) and `UNICODE-ENCODING=8` (for ORW).

For example,

```
Section: User-Source-Tables
*
CREATE_IDT
                        Contact
SOURCED_FROM FLAT_FILE
            Alias         W(32),
            BirthDate     W(20),
            FirstName     W(32),
            LastName      W(32),
            MiddleName    W(32),
            City          W(32),
            Country       W(32),
            PostalCode    W(10),
            StreetAddress W(100),
(PK)     RowId         W(16)

SYNC REPLACE_DUPLICATE_PK
TXN-SOURCE NSA
;
```

Environment Variables

The following environment variables must be defined in the environment used to start IIR servers and utilities.

SSA_XML_UTF16=1 this variable informs IIR to output UTF-16 encoded Unicode into 'W' columns when converting data extracted from XML documents produced by Siebel. When set to zero it uses UTF-8. The default (when not specified) is UTF-16.

SSA_XML_SIZE this variable specifies the size of the XML parsing buffer (in bytes) of the XS Server.

This should be at least as large as `IDS_PAGE_SIZE * <max bytes per Siebel Msg>`. The former is a Profile Attribute of the `ISSLaunchBuildLoadFile` workflow and the latter is a function of the size and number of fields included in the IO.

SSA_RBNAME this variable specifies the connection string for the Rulebase containing the System. Its format is described in the *Rulebase and Database Names* section of this guide.

Loading Data

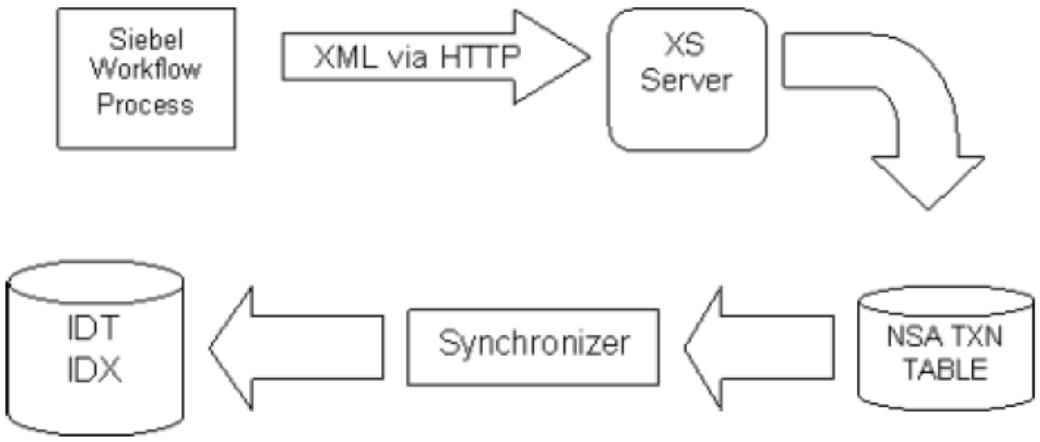
Invoke the IIR supplied Workflow Process named `Launch Build Load File` to extract and write your Siebel data to a XML File, see *Constructing Load Data* section. This file is used as flat-file input for the IIR Table Loader process, see *Table Loader* section.

Synchronization

In order to synchronize IIR with updates to the Siebel application, the Siebel Administrator defines Run- Time Events on the BCs that require synchronization. When the BCs are updated, Run-Time Events invoke Action Sets that subsequently call IIR Workflows to send XML messages to IIR XS Server.

Upon receipt of an XML message, the XS Server parses it to determine the System and IDT that this messages pertains to, and to locate the IO fields that are replicated in the IDT. An IDT record is constructed and stored in the NSA Transaction Table (NSA TT). Refer to the *Update Synchronizer* chapter for details about this table and the synchronization process in general.

Note: The order of message receipt at the XS Server defines the order in which transactions will be processed by the Update Synchronizer.



XS Server

The Siebel HTTP Transport service is used to send XML messages to the IIR XS Server. Unfortunately, the service does not close its connection with the XS Server until the Siebel application user that initiated the connection logs off the Siebel application.

Siebel's failure to close its connections means that the XS Server will not shutdown until all Siebel clients log off.

Restrictions

Siebel Restrictions

- There must be one primary BC per IO.
- An IO may not include any secondary BCs.
- Run-Time Events do not capture batch updates to BCs, leading to a possible loss of synchronization.
- Run-Time Events only trap data changes made by components. Changes made in the Data Manager level will not trigger Run-Time events.
- The Siebel HTTP Transport service does not close its connection to the XS Server until the client (Siebel Application user) logs off their Siebel session.

IIR Restrictions

Transaction Sequence Numbers are generated in the order of XML message receipt (necessary due to the lack of Siebel facilities to generate unique sequence numbers).

- Only one XS Server can be defined to accept XML messages. The use of multiple servers will result in the allocation of duplicate transaction sequence numbers.
- The maximum IDT record length in the NSA Transaction Table is limited by DBMS limits on the size of a binary column.
- UDB Unicode is not currently supported. Only the ASCII subset of UTF-8 can be loaded.

CHAPTER 9

Web Services

This chapter includes the following topics:

- Introduction, 121
- IIR Web Services, 121
- XML Search Service, 122
- XML Console Service, 132
- NSA-Batch Service, 147
- Real Time Web Service, 148
- Custom HTTP Header, 157
- UDDI, 158

Introduction

Web services are software that provide a standard means of interoperating between different applications, running on a variety of platforms over a network. They are characterized by interoperability and extensibility, thanks to the use of XML messages that follow the SOAP standard. They can be combined to produce a Service Oriented Architecture.

IIR Web Services

Identity Resolution provides four web services:

- Search Service
- Console Service
- NSA-Batch Service
- Real-Time Service

This manual describes these web services, and how to use them.

All IIR web services are implemented as free-standing servers rather than as servlets. No web application server like IBM Websphere Application Server (WAS), Microsoft BizTalk or Apache Tomcat is required.

SOAP

All IIR web services use the Simple Object Access Protocol (SOAP). Both SOAP version 1.1 and SOAP version 1.2 are supported. A SOAP 1.1 request will receive a SOAP 1.1 response from the IIR web services; conversely, a SOAP 1.2 request will receive a SOAP 1.2 response.

Unicode

All IIR web services use Unicode. Messages may be sent in UTF-8 or UTF-16. Responses will use the same character set as the original request.

XML Search Service

Deploying the XML Search Service

Identity Resolution provides a web-based XML Search Service. The service is implemented by the XML Search Server, as part of the `ssasrsv` executable image.

Enabling

The XML Search Server does not start unless it has been enabled and configured. The XML Search Server is enabled by allocating the server's host name (`SSA_XMHOST`) and port number (`SSA_XMPORT`) in the `env\isss.bat` (Windows) or `env/isss` (UNIX). The default port number of the XML Search Service Server is 1670.

Configuring

The configuration process consists of creating a simple text file named either `xmserv.ini` or `xmserv.xml`. The two different extensions represent two different formats that the configuration file can take; an INI file form and an XML form.

The file can be located in `$SSAINI`, `$HOME` or `$SSABIN`, which the server searches in that order.

The content of this file determines which searches and Rulebases are visible to the client. It is read at server initialization, so changes to the configuration become effective only after the XML Search Server is restarted or refreshed.

The `xmserv.ini` form has the same format as the htserv.ini file used by the HTTP Search Server. Refer to the *HTTP Search Client* section of the *OPERATIONS Guide* for instructions on how to use this format.

Since this is a Web Service, the XML format is recommended.

Generic Mode

The simplest possible file contains the following tags:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/xmlserv">
<mode>generic</mode>
<rulebase>ids:rb</rulebase>
</server>
```

Use the simple `xmserv.xml` file to make all searches in the `Rulebase ids:rb` available.

Unlike the HTTP Search Server, you must specify a Rulebase for the XML Search server.

Note: Rulebase names are sent from the client to the server in the clear using the HTTP protocol. To avoid passing database passwords, it is strongly recommended that `xmserv.xml` files should use `Dictionary Alias` names. If you did not, the same file would look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/xmlserv">
<mode>generic</mode>
<rulebase>odb:0:userid/topsecretpassword@dbserver6</rulebase>
</server>
```

Custom Mode

Use the custom mode to configure the systems, searches and Rulebases for the Web clients. You can define one or more profiles. Each profile contains rules that define the Rulebase name, system name, and one or more search definitions.

You can use the `sdf_view` parameter to specify the output view that you define in the SDF file for each search definition. A SOAP response might contain different fields for the matching records based on the output views that you specify in the `xmserv.xml` file.

The following sample `xmserv.xml` file defines a single profile in the custom mode:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/xmlserv">
   <mode>custom</mode>
   <profile name="search profile">
      <rule name="search rule">
         <rulebase>ids:rb</rulebase>
         <system>ssa001</system>
         <search name="search name">
            <sdf_search>name-search</sdf_search>
            <sdf_view>View2</sdf_view>
         </search>
      </rule>
   </profile>
</server>
```

The following sample `xmserv.xml` file defines multiple profiles in the custom mode:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/xmserv">
   <mode>custom</mode>
   <profile name="Profile 1">
      <rule name="rule2">
         <rulebase>ids:rb2</rulebase>
         <system>ssa001</system>
         <search name="Name Search">
            <sdf_search>name-search</sdf_search>
            <sdf_view>View2</sdf_view>
         </search>
      </rule>
   </profile>
   <profile name="Profile 2">
      <rule name="rule3">
         <rulebase>ids:rb3</rulebase>
         <system>ssa001</system>
         <search name="Name Search">
            <sdf_search>name-search</sdf_search>
            <sdf_view>View3</sdf_view>
         </search>
      </rule>
      <rule name="rule1">
         <rulebase>ids:rb</rulebase>
         <system>ssa001</system>
         <search name=" Name Search">
            <sdf_search>name-search</sdf_search>
            <sdf_view>View1</sdf_view>
```

```
          </search>
        </rule>
      </profile>
    </server>
```

In the preceding example, you have three rules for three systems, named ssa001, and probably identical, but perhaps residing in three different Rulebases. In this case, four WSDL files will be generated, called `rule1.wsdl`, `rule2.wsdl`, `rule3.wsdl` and `ssa001.wsdl`. The `ssa001.wsdl` file will correspond to `rule1`. Each will have its own target namespace. That of `ssa001.wsdl` will be `http://www.identitysystems.com/xmlschema/iss-version-8/searchSvc` while `rule1.wsdl`, `rule2.wsdl`, and `rule3.wsdl` will use the following namespaces:

```
http://www.identitysystems.com/xmlschema/iss-version-8/searchSvc/rule1
http://www.identitysystems.com/xmlschema/iss-version-8/searchSvc/rule2
http://www.identitysystems.com/xmlschema/iss-version-8/searchSvc/rule3
```

Note: Two rules cannot have the same name. The server will issue an error.

WSDL File

WSDL files are created in the server work directory for each rule and system defined in the `xmserv.xml` file when the server starts or is refreshed.

The WSDL can also be accessed through the server at:

```
http://<xmhost>:<xmport>/?<system>.wsdl
```

Which will correspond to the last-named system of that name in the `xmserv.xml` file. For example, the sample system will usually be found at:

```
http://localhost:1670/?ssa001.wsdl
```

The WSDL can also be retrieved from:

```
http://<xmhost>:<xmport>/?<rule>.wsdl
```

Re-generating the WSDL File

The WSDL file can be regenerated by issuing a flush command to the server. The server will re-read the `xmserv.xml` file and re-create the WSDL file. On a Unix platform this would be done by:

```
$SSABIN/ssashut -h$SSA_XMHOST -f
```

Or on Windows:

```
%SSABIN%\ssashut -h%SSA_XMHOST% -f
```

Note: If a system or search is deleted, it should be manually removed from the `xmserv.xml` file and a flush command should be issued to the server to remove the corresponding web service.

Note: Searches are cached. If a system is modified, a flush command should be issued to the server to regenerate the WSDL file and flush the search cache.

Creating a .NET Proxy

A proxy can be created from the WSDL generated by the XML Search Server using `wsdl.exe`, which is part of the Microsoft .NET SDK. Given the WSDL created from the sample system SSA001 (which can also be found in the `IIR samples\programs\csharp-xml` directory), one can create a proxy with:

```
wsdl /out:ssa001.cs ssa001.wsdl
```

This creates a C# `public class` called IDT001.

```
public class IDT001 {
                            public int score;
                            public string ID;
```

```
                                public string Name;
                                public string DOB;
                                public string Address;
    }
```

This can be then be compiled with: `csc /target:library /out:ssa001.dll ssa001.cs`

and linked with a client like `ws-sample1.cs` in the `IIR samples\programs\csharp-xml` directory.

```
csc /target:exe /reference:ssa001.dll ws-sample1.cs
```

The samples can be built with the supplied `compile.bat` script. If you have Microsoft Web Service Extensions (WSE) 3.0 installed, you may prefer to compile with that instead. The script accepts an argument that instructs it to use WSE 3.0:

```
compile wse3
```

At the heart of the `ws-sample1.cs` sample is:

```
    try {
                    ssa001 search = new ssa001 ();
                    IDT001[] results = search.namesearch (
                                name,address, dob,
                                null, null, null, workdir,
                                search_width, match_tolerance);
                    foreach (IDT001 idt in results) {
                                Console.WriteLine ("{0} {1,-24} {2}",
                                            idt.score, idt.Name, idt.Address);
                                            }
        } catch (SoapException se) {
                    Console.Error.WriteLine (se.Message);
        } catch (WebException we) {
                   Console.Error.WriteLine (we.Message);
        }
```

From this, we can see that:

- The search class has the name of the IIR system.
- The response class has that of the IDT defined in the IIR system.
- The search class contains search methods, which bear the names of the searches defined in the system.
- The searches take parameters which are the fields of the search, plus four options (see below).
- In every case, you can get the default by passing a `null` parameter to the method.
- Errors are thrown as `SOAPException` exceptions.
- There is also the possibility of a `WebException` exception, which may occur if you try to run the client without bringing the server up.

Optional parameters:

LOGOUT

Filename for server output for this session.

LOGERR

Filename for server errors for this session.

LOGTEST

Filename for server search trace for this session.

WORKDIR

Used to inform the Search Server which directory is to be used as the working directory for this session.

Search_width

Specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

Match_tolerance

Specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates

Apache Axis2

An Apache Axis2 sample called `Axis2Sample.java` is included with the Java samples. If you have Apache Axis2 installed, and you paths and classpaths set up correctly, you can build a proxy with:

```
wsdl2java -uri ssa001.wsdl -d adb -s -p ssa001
```

Then compile it with:

```
javac ssa001/Ssa001Stub.java ssa001/Ssa001Fault.java
```

And compile the sample with:

```
javac Axis2Sample.java
```

Running the Samples

To get the samples to run you need to load the sample system and create an `xmserv.xml` file similar to the simple example above.

SOAP Request

The `ws-sample1.cs` sample program will generate a SOAP 1.1 request that will look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soap:Body>
      <name-search xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/
searchSvc" WORKDIR="d:\a2mi\ids\testx270.dir" search_width="Typical"
match_tolerance="Loose" system_name="ssa001">
         <Name>J Smythe</Name>
         <Address>157 cathy st</Address>
         <DOB>19491231</DOB>
      </name-search>
   </soap:Body>
</soap:Envelope>
```

SOAP Response

The response takes the form of a SOAP envelope with an element in the body with the name of the search followed by "`_response`." This contains a result element named after the IDT, with "`Result`" added which in

turn contains the IDT fields, plus an additional one called "score". All names are exactly as they appear in the System Definition File.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsse="http://
docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-
utility-1.0.xsd">
   <soap:Body>
      <name-search_response xmlns="http://www.identitysystems.com/xmlschema/iss-
version-8/searchSvc">
         <Search01Result>
            <Search01>
               <score>85</score>
               <ID>1617</ID>
               <Name>M J SMITH</Name>
               <DOB>19491018</DOB>
               <Address>4/157 CARTHAGE STREET</Address>
               <CL_ID />
            </Search01>
            <Search01>
               <score>80</score>
               <ID>0000001617</ID>
               <IDS-IDX-IO>00000023</IDS-IDX-IO>
               <IDS-IDT-IO>00000000</IDS-IDT-IO>
               <IDS-KSL-ACCEPTED-COUNT>00000000</IDS-KSL-ACCEPTED-COUNT>
               <IDS-KSL-UNDECIDED-COUNT>00000001</IDS-KSL-UNDECIDED-COUNT>
               <IDS-KSL-REJECTED-COUNT>00000019</IDS-KSL-REJECTED-COUNT>
               <IDS-KSL-TOTAL-COUNT>00000020</IDS-KSL-TOTAL-COUNT>
            </Search01>
            (more...)
         </Search01Result>
      </name-search_response>
   </soap:Body>
</soap:Envelope>
```

The SOAP response might contain different fields for the search results based on the output view that you specify in the xmserv.xml file.

Match Explain API

In addition to the XML Search Web Service, there is also an XML Match Explain Web Service. An XML Match Explain request takes two records, known as the search and file records, and describes the reasons why a match scored what it did. The search and file records have the same format as the search record used by the XML SearchWeb Service.

An XML Match Explain request looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <env:Header xmlns:env="http://www.w3.org/2003/05/soap-envelope">
      <wsa:Action />
      <wsa:MessageID>urn:uuid:534141cc-e1c1-48d0-97f8-a5a3e38244f7</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</
wsa:Address>
      </wsa:ReplyTo>
      <wsa:To>http://host:1665/</wsa:To>
      <wsse:Security>
         126 Chapter 9: Web Services
         <wsu:Timestamp wsu:Id="Timestamp-57d31233-8456-4920-9dac-cb01cb261861">
            <wsu:Created>2010-03-30T03:58:40Z</wsu:Created>
            <wsu:Expires>2010-03-30T04:03:40Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
```

```
   </env:Header>
   <soap:Body>
      <Explain-name-search xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/
searchSvc" match_tolerance="Loose" system_name="ssa001">
         <Explain-name-search-search>
            <Name>J Smythe</Name>
            <DOB>19491231</DOB>
            <Address>157 cathy st</Address>
         </Explain-name-search-search>
         <Explain-name-search-file>
            <Name>John Smithe</Name>
            <DOB>19491231</DOB>
            <Address>157 cathy st</Address>
         </Explain-name-search-file>
      </Explain-name-search>
   </soap:Body>
</soap:Envelope>
```

The response will look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
www.w3.org/2005/03/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wsswssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd">
   <soap:Header>
      <wsa:MessageID>urn:uuid:631cb7e4-2e05-4e3f-b6a2-e0968da50e91</wsa:MessageID>
      <wsa:Action>name-search</wsa:Action>
      <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
      <wsa:From>
         <wsa:Address>http://host:1665</wsa:Address>
      </wsa:From>
      <wsse:Security>
         <wsu:Timestamp wsu:Id="Timestamp-aacd0899-39ce-473f-b570-0e7d5c373e06">
            <wsu:Created>2010-03-30T04:58:40Z</wsu:Created>
            <wsu:Expires>2010-03-30T05:03:40Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </soap:Header>
   <soap:Body>
      <Explain-name-search_response xmlns="http://www.identitysystems.com/xmlschema/iss-
version-8/searchSvc">
         <Explain-Result Record-Type="0">
            <Explain-Summary Parent-Sequence-Number="0" Sequence-Number="1">
               <Score>92</Score>
               <Decision>A</Decision>
            </Explain-Summary>
         </Explain-Result>
         <Explain-Result Record-Type="1">
            <Explain-Operator Parent-Sequence-Number="1" Sequence-Number="2">
               <Type>03</Type>
            </Explain-Operator>
         </Explain-Result>
         <Explain-Result Record-Type="2">
            <Explain-Purpose Parent-Sequence-Number="2" Sequence-Number="3">
               <Name>Person_Test</Name>
               <Score>92</Score>
               <Decision>A</Decision>
               <Match-Level>Typical</Match-Level>
               <Accept-Limit>89</Accept-Limit>
               <Reject-Limit>70</Reject-Limit>
               <Early-Exit-Taken>false</Early-Exit-Taken>
            </Explain-Purpose>
         </Explain-Result>
         <Explain-Result Record-Type="4">
            <Explain-Method Parent-Sequence-Number="3" Sequence-Number="4">
               <Field-Name>Person_Name</Field-Name>
               <Score>86</Score>
               <Weight>400</Weight>
               <Original-Weight>4</Original-Weight>
```

```
      <Weight-Flag>W</Weight-Flag>
      <Contributed>true</Contributed>
      <Optional>false</Optional>
      <Contribution>49</Contribution>
      <Repeating-Field>false</Repeating-Field>
      <Search-Index-Used>0</Search-Index-Used>
      <File-Index-Used>0</File-Index-Used>
   </Explain-Method>
</Explain-Result>
<Explain-Result Record-Type="5">
   <Explain-Data Parent-Sequence-Number="4" Sequence-Number="5">
      <Type>S</Type>
      <data>John Smithe</data>
   </Explain-Data>
</Explain-Result>
<Explain-Result Record-Type="5">
   <Explain-Data Parent-Sequence-Number="4" Sequence-Number="6">
      <Type>F</Type>
      <data>J Smythe</data>
   </Explain-Data>
</Explain-Result>
<Explain-Result Record-Type="4">
   <Explain-Method Parent-Sequence-Number="3" Sequence-Number="7">
      <Field-Name>Address_Line1</Field-Name>
      <Score>100</Score>
      <Weight>200</Weight>
      <Original-Weight>2</Original-Weight>
      <Weight-Flag>W</Weight-Flag>
      <Contributed>true</Contributed>
      <Optional>true</Optional>
      <Contribution>28</Contribution>
      <Repeating-Field>false</Repeating-Field>
      <Search-Index-Used>0</Search-Index-Used>
      <File-Index-Used>0</File-Index-Used>
   </Explain-Method>
</Explain-Result>
<Explain-Result Record-Type="5">
   <Explain-Data Parent-Sequence-Number="7" Sequence-Number="8">
      <Type>S</Type>
      <data>157 cathy st</data>
   </Explain-Data>
</Explain-Result>
<Explain-Result Record-Type="5">
   <Explain-Data Parent-Sequence-Number="7" Sequence-Number="9">
      <Type>F</Type>
      <data>157 cathy st</data>
   </Explain-Data>
</Explain-Result>
<Explain-Result Record-Type="4">
   <Explain-Method Parent-Sequence-Number="3" Sequence-Number="10">
      <Field-Name>Date</Field-Name>
      <Score>100</Score>
      <Weight>100</Weight>
      <Original-Weight>1</Original-Weight>
      <Weight-Flag>W</Weight-Flag>
      <Contributed>true</Contributed>
      <Optional>true</Optional>
      <Contribution>14</Contribution>
      <Repeating-Field>false</Repeating-Field>
      <Search-Index-Used>0</Search-Index-Used>
      <File-Index-Used>0</File-Index-Used>
   </Explain-Method>
</Explain-Result>
<Explain-Result Record-Type="5">
   <Explain-Data Parent-Sequence-Number="10" Sequence-Number="11">
      <Type>S</Type>
      <data>19491231</data>
   </Explain-Data>
</Explain-Result>
<Explain-Result Record-Type="5">
```

```
            <Explain-Data Parent-Sequence-Number="10" Sequence-Number="12">
               <Type>F</Type>
               <data>19491231</data>
            </Explain-Data>
         </Explain-Result>
         <Explain-Result Record-Type="1">
            <Explain-Operator Parent-Sequence-Number="1" Sequence-Number="13">
               <Type>04</Type>
            </Explain-Operator>
         </Explain-Result>
      </Explain-name-search_response>
   </soap:Body>
</soap:Envelope>
```

Refer to the *Match Explain API* section of the *DEVELOPER GUIDE* for a description of the meanings of these fields.

Web Services Addressing

IIR Web Services supports Web Services Addressing.

Web Services Addressing Standards

IIR Web Services supports the Web Services Addressing 1.0 - Core W3C Recommendation dated 9 May 2006,Web Services Addressing 1.0 - SOAP Binding W3C Recommendation dated 9 May 2006, and Web Services Addressing 1.0 - WSDL Binding W3C Candidate Recommendation dated 29 May 2006.

To deploy this facility on, start the servers by running the shell script $SSABIN/idsup on Unix or the batch script %SSABIN%\idsup.bat on Windows with the following option:

-qca1.0 Specifies that `WS-Addressing 1.0` will be used

With `WS-Addressing`, a request will be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <env:Header xmlns:env="http://www.w3.org/2003/05/soap-envelope">
      <wsa:Action />
      <wsa:MessageID>urn:uuid:ec859556-55c3-4256-83bc-e134902f1323</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</
wsa:Address>
      </wsa:ReplyTo>
      <wsa:To>http://host:1670/</wsa:To>
      <wsse:Security>
         <wsu:Timestamp wsu:Id="Timestamp-0d7269e6-691f-4539-bab8-a44677b78d00">
            <wsu:Created>2009-07-10T00:16:50Z</wsu:Created>
            <wsu:Expires>2009-07-10T00:21:50Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </env:Header>
   <soap:Body>
      <name-search xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/
searchSvc" WORKDIR="d:\a2mi\ids\testx270.dir" search_width="Typical"
match_tolerance="Loose" system_name="ssa001">
         <Name>J Smythe</Name>
         <Address>157 cathy st</Address>
         <DOB>19491231</DOB>
      </name-search>
   </soap:Body>
</soap:Envelope>
```

With `WS-Addressing` switched on, the servers will require a valid `WS-Addressing` header to be present.

Note: The servers will validate the security timestamp, if present. You may therefore need to ensure that your server machine's clock is accurate.

The response will be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/03/addressing" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd" xmlns:wsu="http://
docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd">
   <soap:Header>
      <wsa:MessageID>urn:uuid:38bfb6b5-ce10-4768-aaa9-852d1e55605f</wsa:MessageID>
      <wsa:Action>name-search</wsa:Action>
      <wsa:To>http://www.w3.org/2005/03/addressing/role/anonymous</wsa:To>
      <wsa:From>
         <wsa:Address>http://host:1670</wsa:Address>
      </wsa:From>
      <wsse:Security>
         <wsu:Timestamp wsu:Id="Timestamp-86d8102a-ed7f-4bea-bee1-dd17b8dc954a">
            <wsu:Created>2009-07-10T01:00:31Z</wsu:Created>
            <wsu:Expires>2009-07-10T01:05:31Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </soap:Header>
   <soap:Body>
      <name-search_response xmlns="http://www.identitysystems.com/xmlschema/iss-
version-8/searchSvc">
         <Search01Result>
            <Search01>
               <score>85</score>
               <ID>1617</ID>
               <Name>M J SMITH</Name>
               <DOB>19491018</DOB>
               <Address>4/157 CARTHAGE STREET</Address>
               <CL_ID />
            </Search01>
            <Search01>
               <score>70</score>
               <ID>1647</ID>
               <Name>JACK SMITH</Name>
               <DOB>19201220</DOB>
               <Address>22 BRUCE STREET</Address>
               <CL_ID />
            </Search01>
            (more...)
         </Search01Result>
      </name-search_response>
   </soap:Body>
</soap:Envelope>
```

Web Services Security

IIR Web Services supports Web Services Security.

Web Services Security Standards

IIR Web Services supports the Web Services Security SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification dated 1 February 2006.

Samples

The sample `ws-sample3.cs`, which requires WSE 3.0, uses a supplied sample X509 RSA certificate to create a message which is signed with private key. A copy of the X509 RSA certificate is included in the message.

The IIR servers use the supplied public RSA key to validate the request.

Transport Layer Security

IIR web services can employ HTTPS to implement Transport Layer Security. This will provide point to point security. To deploy this facility on, start the servers by running the shell script $SSABIN/idsup on Unix or the batch script %SSABIN%\idsup.bat on Windows with the following options:

-qcFile1

Specifies the PEM file containing an X509 certificate.

-qkFile2

Specifies the PEM file containing an RSA private key.

-qrFile3

Specifies the PEM file containing an X509 root certificate.

The web service will now use HTTPS instead of `HTTP. HTTPS` sends HTTP messages using SSL, a well established and widely available security protocol. If HTTPS is specified, any messages sent to the web service using HTTP will be discarded.

Note: All three options must be specified. The server will report an error on startup if one is omitted.

Timeout Period for XML Search Server

You can specify the timeout periods as a comma-separated list for the XML Search Server when you start the Identity Resolution Server.

To specify the timeout periods when you start the Identity Resolution Server, perform the following task:

- On Windows: On a command line, run the `idsup.bat -qt"<Idle>,<Connection>,<Read>,<Write>"` command from the `<Identity Resolution Installation Directory>\bin` directory.
- On UNIX: On a shell command line, run the `idsup -qt<Idle>,<Connection>,<Read>,<Write>` command from the `<Identity Resolution Installation Directory>/bin` directory.

The `idsup` command uses the following parameters:

Idle

A client session to remain idle before the server cancels the session. Specify the timeout period in seconds. Set the value as 0 if you want to use the default time period.

Connection

A connection to the server to wait before terminating the attempt. Specify the timeout period in seconds. Set the value as 0 if you want to use the default time period.

Read

A read or receive operation to complete successfully. Specify the timeout period in seconds. Set the value as 0 if you want to use the default time period.

Write

A write or send operation to complete successfully. Specify the timeout period in seconds. Set the value as 0 if you want to use the default time period.

XML Console Service

IIR provides a web based XML Console Service. The service is implemented by the XML Console Server, as part of the `ssacssv` executable image.

Enabling the XML Console Service

The XML Console Server will not start unless it has been enabled.

The XML Console Server is enabled by allocating the server's host name (`SSA_CXHOST`) and port number (`SSA_CXPORT`) in the `env\isss.bat` (Windows) or `env/isss` (UNIX). The default port number of the XML Console Server is `1673`.

WSDL file

A WSDL file is created in the server work directory when the server starts or is refreshed. The WSDL can also be accessed through the server at:`http://<cshost>:<csport>/?console.wsdl`

For example, the sample system will usually be found at: `http://localhost:1673/?console.wsdl`

Creating a .NET Proxy

A proxy can be created from the `console.wsdl` using `wsdl.exe`, which is part of the Microsoft .NET SDK.

```
wsdl /out:console.cs console.wsdl
```

XML Console Service Functions

The XML Console server provides Web Services with some of the features of the IIR Console, although it does not provide all the functionality. The following Console functions are supported:

Web Service	Description	Parameters	Return Code
ssacx_connect	Initiates a socket.	host is the host to connect to. port is the port to connect to. sockh is a socket handle.	negative for error, 0 for success

Input message

```
<?xml version="1.0" ?>
<soap:Envelope
xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_connect
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<host>value</host>
<port>301</por t>
</ssacx_connec t>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0" ?>
<soap:Envelope
xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_connect_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
<sockh>302</sockh>
</ssacx_connec t_response>
```

```
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_database_create	Create a new database.	`database_name` - The name of the database. `work_directory` - The work directory for the server to use.	negative for error, 0 for success

Input message

```
<?xml version=" 1 . 0 " ?>
<soap:Envelope
xmlns:soap=" h t t p ://schemas . xmlsoap . org/soap/envelope/">
<soap:Body>
<ssacx_database_create
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<database_name>value</database_name>
<work_directory>value</work_directory>
</ssacx_database_create>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_database_create_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_database_create_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_database_delete	Delete an existing database.	`database_name` - The name of the database. `work_directory` - The work directory for the server to use.	negative for error, 0 for success

Input message

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_database_delete
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<database_name>value</database_name>
<work_directory>value</work_directory>
</ssacx_database_delete>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0"?>
```

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_database_delete_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_database_delete_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_disconnect	Releases resources allocated to a socket.	none	negative for error, 0 for success

Input message

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_disconnect
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc "></
ssacx_disconnect>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_disconnect_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_disconnect_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_errors_get_all	Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.	msg is an error message.	negative for error, 0 for success

Input message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_errors_get_all
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<msg_size>256</msg_size>
</ssacx_errors_get_all>
```

```
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_errors_get_all_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
<msg>value</msg>
</ssacx_errors_get_all_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_job_run	Run a user job. Start the job and report status when it is finished.	`rulebase_name` - The name of the rulebase. `system_name` - The name of the system `job_name` -The name of the job `job_number`- The number of the job `job_report`- The report on the progress of the job `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_run
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<job_name>value</job_name>
<job_report_size>256</job_report_size>
<work_directory>value</work_directory>
</ssacx_job_run>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_run_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
<job_number>302</job_number>
<job_ report>value</job_ report>
</ssacx_job_run_response>
```

```
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_job_start	Start a user job and return immediately.	`rulebase_name` - The name of the rulebase. `system_name` - The name of the system `job_name` -The name of the job `job_number`- The number of the job `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message

```
<?xml version="1.0" ?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<job_name>value</job_name>
<work_directory>value</work_directory>
</ssacx_job_start>
</soap:Body>
</soap:Envelope>
```

Output message

```
<?xml version="1.0" ?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
<job_number>302</job_number>
</ssacx_job_start_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_job_status	Get the status of a user job.	`rulebase_name` - The name of the rulebase. `system_name` - The name of the system `job_name` -The name of the job `job_number`- The number of the job `job_status`-The status of the job (0 = success, 1 = running, -ve = failed) `job_report`- The report on the progress of the job `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0" ?>
<soap:Envelope
```

```
xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_status
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<job_name>value</job_name>
<job_number>301</job_number>
<job_report_size>256</job_report_size>
<work_directory>value</work_directory>
</ssacx_job_status>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_status_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
<job_status>0</job_status>
<job_report>value</job_report>
</ssacx_job_status_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_job_stop	Stop a user job.	`rulebase_name` - The name of the rulebase. `system_name`- The name of the system `job_number`- The number of the job `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_stop
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<job_number>301</job_number>
<work_directory>value</work_directory>
</ssacx_job_stop>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_job_stop_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_job_stop_response>
```

```
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_jobs_running	List all the running jobs	`job_name` The name of the job `job_number` The number of the job `job_stepid` The number of the step inside the job `job_progress` A description of the activity of the job	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_jobs_running
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc"></
ssacx_jobs_running>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0" ?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_jobs_running_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
<jobs_running>
<job_name>value</job_name>
<job_number>302</job_number>
<job_stepid>302</job_ stepid>
<job_progres sArray>
<job_progres s>value</job_progress>
</job_progres sArray>
</jobs_running>
</ssacx_jobs_running_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_rulebase_create	Create a new rulebase.	`rulebase_name` -The name of the rulebase `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_rulebase_create
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<rulebase_name>value</rulebase_name>
<work_directory>value</work_directory>
</ssacx_rulebase_create>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_rulebase_create_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_rulebase_create_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_rulebase_delete	Delete an existing rulebase.	`rulebase_name` -The name of the rulebase `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_rulebase_delete
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<rulebase_name>value</rulebase_name>
<work_directory>value</work_directory>
</ssacx_rulebase_delete>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_rulebase_delete_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_rulebase_delete_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_server_check	Check the status of a server	`address`- The URL of the server	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_check
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<address>value</address>
</ssacx_server_check>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_check_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_server_check_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_server_flush	Issue a flush command to a server	`address`- The URL of the server	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_flush
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<address>value</address>
</ssacx_server_flush>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_flush_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_server_flush_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_server_start	Start a server	`server` -The name of the server (HT = HTTP, RB = rulebase, SE = search, XM = XML search, XS = XML synchronizer) `address`- The URL of the server `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<server>value</server>
<address>value</address>
```

```
<work_directory>value</work_directory>
</ssacx_server_start>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_server_start_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_server_stop	Stop a server	`address`- The URL of the server	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_stop
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<address>value</address>
</ssacx_server_stop>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_server_stop_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc ">
<response>0</response>
</ssacx_server_stop_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_servers_start	Start the servers	none	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_servers_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc"></
ssacx_servers_star</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_servers_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_servers_start_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_servers_stop	Stop the servers	none	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_servers_stop
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc"></
ssacx_servers_stop>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_servers_stop_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_servers_stop_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_system_create	Create a new system from a System Definition File (SDF)	`rulebase_name`- The name of the rulebase `database_name`- The name of the database `system_name`- The name of the system `work_directory`- The work directory for the server to use `sdf_name`- The name of the System Definition File	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_create
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<database_name>value</database_name>
```

```
<system_name>value</system_name>
<work_directory>value</work_directory>
<sdf_name>value</sdf_name>
</ssacx_system_create>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_create_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_system_create_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_system_delete	Delete an existing database.	`rulebase_name`- The name of the rulebase `system_name`- The name of the system `work_directory`- The work directory for the server to use	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_delete
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<work_directory>value</work_directory>
</ssacx_system_delete>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_delete_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_system_delete_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_system_import	Create a system from an export file	rulebase_name- The name of the rulebase database_name- The name of the database system_name- The name of the system work_directory- The work directory for the server to use file_name- The name of the file	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_import
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<database_name>value</database_name>
<system_name>value</system_name>
<work_directory>value</work_directory>
<file_name>value</file_name>
</ssacx_system_import>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_import_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_system_import_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_system_status_get	Get the status of a system	rulebase_name- The name of the rulebase system_name- The name of the system work_directory- The work directory for the server to use system_status- The system status (build, locked, production, test or prototype)	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_status_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<work_directory>value</work_directory>
<system_status_size>256</system_status_size>
</ssacx_system_status_get>
```

```
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_status_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
<system_status>value</system_status>
</ssacx_system_status_get_response>
</soap:Body>
</soap:Envelope>
```

Web Service	Description	Parameters	Return Code
ssacx_system_status_set	Set the status of a system	`rulebase_name`- The name of the rulebase `system_name`- The name of the system `work_directory`- The work directory for the server to use `system_status`- The system status (build, locked, production, test or prototype)	negative for error, 0 for success

Input message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_status_set
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<rulebase_name>value</rulebase_name>
<system_name>value</system_name>
<work_directory>value</work_directory>
<system_status>value</system_status>
</ssacx_system_status_set>
</soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ssacx_system_status_set_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/consoleSvc">
<response>0</response>
</ssacx_system_status_set_response>
</soap:Body>
</soap:Envelope>
```

Re-generating the WSDL file

The WSDL file can be regenerated by issuing a flush command to the server.

On Unix this would be done by:

```
$SSABIN/ssashut -h$SSA_CXHOST -f
```

Or on Windows:

```
%SSABIN%\ssashut -h%SSA_CXHOST% -f
```

NSA-Batch Service

This is a web based Service which provides the ability to add records to the NSA Transaction table. The service is implemented by the Synchronization Server, as part of the `ssasrsv` executable image.

Enabling the NSA-Batch Service

In order for the NSA-Batch Service to be available the Synchronization Server must be running. In addition, the NSA-Batch Service must be configured

The Synchronization Server is enabled by allocating the server's host name (`SSA_XSHOST`) and port number (`SSA_XSPORT`) in the `env\isss.bat` (Windows) or `env/isss` (UNIX).The default port number of the Synchronization Server is `1671`.

Configuring

The configuration process consists of creating a simple text file named either `xsserv.ini` or `xsserv.xml`. The two different extensions represent two different formats that the configuration file can take; an INI file form and an XML form.

The file can be located in `$SSAINI`, `$HOME` or `$SSABIN`, which the server searches in that order.

The content of this file determines which searches and Rulebases are visible to the client. It is read at server initialization, so changes to the configuration become effective only after the XML Search Server is bounced.

The `xsserv.ini` form has the same format as the `htserv.ini` file used by the HTTP Search Server. Refer to the *HTTP Search Client* section of the *OPERATIONS Guide* for instructions on how to use this format.

Since this is a Web Service, the XML format is recommended.

Generic Mode

The simplest possible file contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/xsserv">
<mode>generic</mode>
<rulebase>ids:rb</rulebase>
</server>
```

This simple `xsserv.xml` will make all searches in the `Rulebase ids:rb` available.

Unlike the HTTP Search Server, a Rulebase must be supplied to the XML Search server.

Note: Rulebase names are sent from the client to the server in the clear using the HTTP protocol. To avoid passing database passwords, it is strongly recommended that `xsserv.xml` files should specify Dictionary Alias names.

Custom Mode

Custom mode is use to configure the Systems, Searches and Rulebases visible to the Web clients. A custom `xsserv.xml` file will look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-1/xsserv">
<mode>custom</mode>
<profile name="search profile">
<rule name="search rule">
<rulebase>ids:rb</rulebase>
<system>ssa001</system>
<search name="search name">
<sdf_search>name-search</sdf_search>
</search>
</rule>
</profile>
</server>
```

The example defines one profile but multiple profiles can be defined. Each may contain one or more rules. In this case, there is just one rule. Each rule must have a corresponding definition that nominates the Rulebase name, System name and any number of Searches (`name-search` in this example) that can be used by a client.

WSDL file

A WSDL file is created in the server work directory for each system defined in the `xsserv.xml` file when the server starts or is refreshed.

The WSDL can also be accessed through the server at:

```
http://<xshost>:<xsport>/?<system>sync.wsdl
```

For example, the sample system would be found at:

```
    http://localhost:1670/?ssa001sync.wsdl
```

Re-generating the WSDL file

The WSDL file can be regenerated by issuing a flush command to the server. The server will re-read the `xsserv.xml` file and re-create the WSDL file.

On Unix this would be done by:

```
$SSABIN/ssashut -h$SSA_XSHOST -f
```

Or on Windows:

```
%SSABIN%\ssashut -h%SSA_XSHOST% -f
```

Using the NSA-Batch Service

The NSA-Batch Service accepts an XML file containing a set of IDT records to be synchronized.

Real Time Web Service

This section provides information on real time Web Service.

Enabling the Real Time Web Service

The Real Time Web Service is provided by the Synchronization Server. Therefore, The Synchronization Server must be running. This is achieved by allocating the server's host name (`SSA_XSHOST`) and port number (`SSA_XSPORT`) in `env\isss.bat` (Windows) or `env/isss` (UNIX). The default port number of the Synchronization Server is 1671.

Configuring

The configuration process consists of creating a simple text file named either `xrserv.ini` or `xrserv.xml`. The file extensions represent the two different formats that the configuration file may take.

The file must be located in either `$SSAINI`, `$HOME` or `$SSABIN`. These 3 directories are searched in that order. If the configuration file is not found, then the Real Time Web Service will not be available.

The contents of the configuration file determine which IDTs will be synchronized by the Service.

Note: The configuration file is read at server initialization time, so changes to the configuration become effective only after the server has been restarted.

The `xrserv.ini` form has the same format as the `htserv.ini` file used by the HTTP Search Server. Refer to the *HTTP Search Client* section of the *OPERATIONS Guide* for instructions on how to use this format.

Since this is a Web Service, the XML format is recommended.

Configuration Settings

You can specify the following settings in the `xrserv.xml` file.

mode

A required global parameter. Possible values are generic or custom. Generic indicates that all the synchronized IDTs in the specified rulebase must be made available to Real Time clients. Custom indicates that only the specified IDTs must be made available to Real Time clients.

txn_thread_num

An optional global parameter. Specifies the number of threads that you want to set to process the IDT updates.

work_queue_size

An optional global parameter. Specifies the size of the IDT or IDX transaction reader input queue. The default size is 5000.

txn_commit_rate

An optional global parameter. Select an appropriate commit rate by tuning. In general, a high commit rate provides better transaction throughput. However, too high a rate might cause the database to run out of rollback space in a multi-user update environment, and updated records might not be visible to searches for long periods. The default value is 1.

rulebase

Required. Specifies the rulebase to use for Real Time operations. Specify at the global level for the generic mode or at the rule level for the custom mode.

system

A required rule-level parameter. Specifies the Identity Resolution system containing the IDT to be processed.

idt

A required rule-level parameter. Specifies the name of the IDT to be processed. This IDT must be present in the specified Identity Resolution system.

disable_idt_mutex_locking

An optional rule-level parameter. Disables the IDT mutex lock and improves synchronization performance. The default value is false.

Note: If you are using a persistent identifier (PID), do not change the default value of the `disable_idt_mutex_locking` parameter.

del_not_found_warn_only

Changes the behaviour to return only a warning when the IDT record cannot be found for a delete transaction.

soft_skip_delete_fail

Skips restarting the Synchronization Server when a record that you try to delete does not exist.

skip_error_and_continue

Skips any errors and continues to process other records.

Generic Mode

The simplest possible file contains the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/xrserv">
    <mode>generic</mode>
    <rulebase>ids:rb</rulebase>
</server>
```

Sample `xrserv.xml` with worker threads configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/xrserv">
    <mode>generic</mode>
    <txn_thread_num>nn</txn_thread_num>
    <rulebase>ids:rb</rulebase>
</server>
```

Note: By default, the number of worker threads (nn) is set to the number of CPUs available on the machine. You can override this value by setting `<txn_thread_num>`.

This simple `xrserv.xml` can synchronize all system and its IDTs in `Rulebase ids:rb`. Rulebase name must be specified.

Note: Rulebase names are sent from the client to the server in the clear using the HTTP protocol. To avoid passing database passwords, Informatica Corporation recommends that the `xsserv.xml` files must specify Dictionary Alias names.

Custom Mode

You can use the Custom mode to restrict the set of IDTs which can be synchronized by using the Real Time Web Service. A sample custom `xrserv.xml` file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns= "http://www.identitysystems.com/xmlschema/iss-version-1/xrserv">
<mode>custom</mode>
<txn_thread_num>nn</txn_thread_num>
      <profile name="IDT001">
         <rule name ="RULEIDT001">
             <rulebase>ids:rb</rulebase>
             <system>ssa001</system>
             <idt>IDT001</idt>
         </rule>
      </profile>
</server>
```

This example defines one profile but multiple profiles can be defined. Each may contain one or more rules. In this case, there is just one rule. Each rule must have a corresponding definition that nominates the Rulebase name, System name and any number of IDTs (IDT001 in this example) that can be synchronized.

Sequence Numbers

The Synchronization Server can process updates from many sources at once. To ensure updates to the same IDT record are performed in the desired order sequence numbers are used. The Synchronization Server conforms to the same format for sequence numbers as NSA transaction table. That is a 32 byte string. This means that string comparison are used for ordering purposes. If using numbers they should be right justified, zero filled. We recommend using a single pool of sequence numbers for every IDT.

Operation Types

You must specify the type of operation that you want to perform in the SOAP request. Use the `IDS_OP` parameter to specify the type of operation.

You can perform the following operations on the records:

- `A`. Adds the record.
- `D`. Deletes the record.
- `S`. Shuts down the Synchronization Server.
- `T`. Forces the Synchronization Server to shut down.
- `M`. Starts the maintenance period during which the Synchronization Server does not process any records.
- `N`. Ends the maintenance period, and the Synchronization Server starts processing the records.

Real Time Reject table layout

The Real Time Web Service stores rejected transactions in an SQL-accessible table named `IDS_UPD_SYNC_REJ`. The layout of the table is as follows:

System Name

The name of the System to which these rejected transactions belong.

IDT Name

The name of the IDT that this record belongs to. This is the fully decorated table name as it appears on the target database. For example an IDT named IDT-99 in the definition file stored on dbid 01, would be called `IDS_01_IDT_99`.

Operation Code

Defines the operation to be applied. Valid values are 'A' meaning add this IDT record, and 'D' meaning delete this IDT record.

IDT Record

This is the rejected record.

Rejected Time Stamp

An alphanumeric string containing the date and time the record was rejected. Format is `YYYYMMDDHHMMSS`.

Note: The `FLAT-FILE` input will not insert record into reject table if `TXN` sequence number is old.

Real Time Failure on System Refresh and Delete

It is not possible to refresh or delete a system while an IDT which belongs to that System is available for Real Time Synchronization. Before these operations can be performed, the active `xrserv.xml` file must be deleted (or moved to another location) and the Synchronization Server must be flushed using the following command:

For Windows:

```
%SSABIN%\ssashut -h%SSA_XSHOST% -f
```

For Unix

```
$SSABIN/ssashut -h$SSA_XSHOST -f
```

Deploying a Real-Time Web Service

This section outlines the steps required to establish a Real-Time Web Service in an IIR environment.

Prerequisites

The Synchronization Server must be running and the Real-Time Web Service must be configured. See the *Enabling the Real Time Web Service* section for further details.

WSDL

WSDL files are created by the Synchronization Server in the server work directory for each rule and system defined in the `xrserv.xml` file when the server starts or is refreshed. The WSDL can also be accessed through the server at:

```
http://<xshost>:<xsport>/?<system>.wsdl
```

where `<system>` is the last-mentioned system in the `xrserv.xml` file. For example, the `ssa003` sample system will usually be found at:`http://localhost:1670/?testx345.wsdl`

The WSDL can also be retrieved from :`http://<xshost>:<xsport>/?<rule>.wsdl`

Re-generating the WSDL file

The WSDL file can be regenerated by issuing a flush command to the server. The server will re-read the `xrserv.xml` file and re-create the WSDL file. For example, on Windows:

```
%SSABIN%\ssashut -h%SSA_XSHOST% -f
```

Or on Unix:

```
$SSABIN/ssashut -h$SSA_XSHOST -f
```

Creating a .NET Proxy

One can create a C# .NET proxy with the supplied compile.bat script in the csharp-xml directory. You must have Microsoft Web Service Extensions (WSE) 3.0 installed. The script accepts an argument that instructs it to use WSE 3.0. This is required by `ws-sample6.cs`, so compile with: `compile wse3`

The first step is to build the proxy:`%ProgramFiles%\Microsoft WSE\v3.0\Tools\WseWsdl3.exe /out:ssa003.cs ssa003.wsdl`

This can be then be compiled with: `csc /target:library /out:ssa003.dll ssa003.cs`

and linked with a client program, `csc /target:exe /reference:ssa003.dll ws-sample6.cs`

The main part of the `ws-sample6.cs` is:

```
ssa003 sync = new ssa003 ();
if (null != xrserv)
                            sync.Url = xrserv;

int response = sync.IDT003 (AcctNo, Address,
                        CL_ID, DOB, Name, IDS_OP, IDS_SEQ,
                        out status, out messages);

foreach (AuditMSG message in messages) {
                                Console.WriteLine ("response={0}", response);
                                Console.WriteLine ("status={0}", status);
                                Console.WriteLine ("rulebase={0}", message.Rulebase);
                                Console.WriteLine ("system={0}", message.System);
                                Console.WriteLine ("IDT={0}", message.IDT);

foreach (ClusterAction cluster_action in message.ClusterAction) {
                                Console.WriteLine ("Type={0}", cluster_action.Type);
                                Console.WriteLine ("ID={0}", cluster_action.To.ID);
```

```
                                    Console.WriteLine ("No={0}", cluster_action.To.No);
                                    }
    }
```

From this, we can see that

- The sync class has the name of the IIR system.
- The method `sync.IDT345` takes IDT input fields to be synchronized as parameters.

Extract from the SDF File

```
Section: User-Source-Tables
*
create_idt
                                    idt003
sourced_from flat_file
                                    (pk)          AcctNo                  C(11),
                                                  Name                       C(50),
                                                     DOB                        C(8),
                                                     Address              C(40)

SYNC REPLACE_DUPLICATE_PK
TXN-SOURCE NSA
```

Layout of the IDT

Column
RECD
IDS_ACCTNO
IDS_NAME
IDS_STREET
IDS_ADDR4
IDS_CITY
IDS_STATE
IDS_ZIP
IDS_GEOALPHA
IDS_LASTNAMECHAR2
IDS_CL_ID

- The response message includes TXN processing status, response, cluster-related information etc.
- Errors are thrown as SOAPException exceptions.
- WebException exceptions may be thrown. This will occur if an attempt is made to run the client without bringing the server up.

Example Soap Messages

Use the real-time web service to perform add, update, and delete operations.

The following examples show the input and output messages when you perform add, update, and delete operations:

Add Operation: Input Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        <wsu:Created>2011-07-01T07:16:13.938Z</wsu:Created>
        <wsu:Expires>2011-07-01T07:17:53.938Z</wsu:Expires>
        </wsu:Timestamp>
</wsse:Security>
<wsa:To>http://d-xpx86-ross:1670</wsa:To>
<wsa:MessageID>urn:uuid:7E1A0F1206D156B17A1309504573377</wsa:MessageID>
<wsa:Action>http://www.identitysystems.com/xmlschema/iss-version-8/RealTimeSync/Sync/
ssa003/IDT003</wsa:Action>
</soapenv:Header>
<soapenv:Body>
<ns1:IDT003 xmlns:ns1="http://www.identitysystems.com/xmlschema/iss-version-8/
RealTimeSync" IDS_OP="A" IDS_SEQ="<ns1:AcctNo>144</ns1:AcctNo>
<ns1:Address>9 TORRENS STREET</ns1:Address>
<ns1:CL_ID/>
<ns1:DOB>19661017</ns1:DOB>
<ns1:Name>MARSHALL ROBERT</ns1:Name>
</ns1:IDT003>
</soapenv:Body>
</soapenv:Envelope>
```

Add Operation:Output Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
www.w3.org/2005/08/addressing
<soap:Header>
<wsa:MessageID>urn:uuid:136df11a-eaab-4d3b-89bb-5e0deab55396</wsa:MessageID>
<wsa:Action>http://www.identitysystems.com/xmlschema/iss-version-8/RealTimeSync/Sync/
ssa003/IDT003</wsa:Action>
<wsa:RelatesTo wsa:RelationshipType="http://www.w3.org/2005/08/addressing/
reply">urn:uuid:7E1A0F1206D156B17A1309504573377</<wsa:To>http://www.w3.org/2005/08/
addressing/role/anonymous</wsa:To>
<wsa:From>
            <wsa:Address>http://d-xpx86-ross:1670</wsa:Address>
</wsa:From>
<wsse:Security>
        <wsu:Timestamp wsu:Id="Timestamp-4ca940fc-b215-4c99-8b45-e6aede5be831">
        <wsu:Created>2011-07-01T07:16:14Z</wsu:Created>
        <wsu:Expires>2011-07-01T07:21:14Z</wsu:Expires>
        </wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body>
    <ssa:IDT003_response xmlns:ssa="http://www.identitysystems.com/xmlschema/iss-
version-8/RealTimeSync">
<ssa:response>0</ssa:response>
<ssa:status>1</ssa:status>
<ssa:AuditMSG>
<ssa:System>ssa003</ssa:System>
<ssa:IDT>IDT003</ssa:IDT>
<ssa:Rulebase>ids:rb</ssa:Rulebase>
<ssa:ClusterAction>
<ssa:Type>Delete from Cluster</ssa:Type>
<ssa:To>
<ssa:ID>AA</ssa:ID>
<ssa:No>4683</ssa:No>
</ssa:To>
<ssa:New>false</ssa:New>
```

```
<ssa:Automatic>true</ssa:Automatic>
</ssa:ClusterAction>
</ssa:AuditMSG>
<ssa:AuditMSG>
<ssa:System>ssa003</ssa:System>
<ssa:IDT>IDT003</ssa:IDT>
<ssa:Rulebase>ids:rb</ssa:Rulebase>
<ssa:ClusterAction>
<ssa:Type>Add to Cluster</ssa:Type>
<ssa:To>
<ssa:ID>AA</ssa:ID>
<ssa:No>4683</ssa:No>
</ssa:To>
<ssa:New>false</ssa:New>
<ssa:Automatic>true</ssa:Automatic>
</ssa:ClusterAction>
</ssa:AuditMSG>
</ssa:IDT003_response>
</soap:Body>
</soap:Envelope>
```

Update Operation: Input Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <env:Header xmlns:env="http://www.w3.org/2003/05/soap-envelope">
      <wsa:Action>http://www.identitysystems.com/xmlschema/iss-version-11/RealTimeSync/
Sync/testx542/IDT542</wsa:Action>
      <wsa:MessageID>urn:uuid:fa128f7a-b243-4b83-89f8-c3e90740b4a3</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</
wsa:Address>
      </wsa:ReplyTo>
      <wsa:To>http://d-xpx86-ross:1670/</wsa:To>
      <wsse:Security>
         <wsu:Timestamp wsu:Id="Timestamp-88962cab-bf9f-45a8-bac7-a7440671fa16">
            <wsu:Created>2015-05-21T03:09:05Z</wsu:Created>
            <wsu:Expires>2015-05-21T03:14:05Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </env:Header>
   <soap:Body>
      <IDT542 xmlns="http://www.identitysystems.com/xmlschema/iss-version-11/
RealTimeSync" IDS_OP="U" IDS_SEQ="3001634343501" user="DXXrross">
         <AcctNo>5835475</AcctNo>
         <Address>5 Torrens</Address>
         <CL_ID />
         <DOB>19660710</DOB>
         <Name>Robert Baratheon</Name>
      </IDT542>
   </soap:Body>
</soap:Envelope>
```

Update Operation:Output Message

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
www.w3.org/2005/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <soap:Header>
      <wsa:MessageID>urn:uuid:8ef2ec72-2e27-40b4-92a4-34b084236a10</wsa:MessageID>
      <wsa:Action>http://www.identitysystems.com/xmlschema/iss-version-11/RealTimeSync/
Sync/testx542/IDT542</wsa:Action>
      <wsa:RelatesTo wsa:RelationshipType="http://www.w3.org/2005/08/addressing/
reply">urn:uuid:fa128f7a-b243-4b83-89f8-c3e90740b4a3</wsa:RelatesTo>
      <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
```

```
      <wsa:From>
         <wsa:Address>http://D-XPX86-ROSS:1670</wsa:Address>
      </wsa:From>
      <wsse:Security>
         <wsu:Timestamp wsu:Id="Timestamp-26d8b1b2-2258-44cc-9bf5-c2f3c8cf9d30">
            <wsu:Created>2015-05-21T03:09:05Z</wsu:Created>
            <wsu:Expires>2015-05-21T03:14:05Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </soap:Header>
   <soap:Body>
      <ssa:IDT542_response xmlns:ssa="http://www.identitysystems.com/xmlschema/iss-
version-11/RealTimeSync">
         <ssa:response>0</ssa:response>
         <ssa:status>1</ssa:status>
         <ssa:AuditMSG>
            <ssa:System>testx542</ssa:System>
            <ssa:IDT>IDT542</ssa:IDT>
            <ssa:Rulebase>ids:rb</ssa:Rulebase>
         </ssa:AuditMSG>
         <ssa:AuditMSG>
            <ssa:System>testx542</ssa:System>
            <ssa:IDT>IDT542</ssa:IDT>
            <ssa:Rulebase>ids:rb</ssa:Rulebase>
         </ssa:AuditMSG>
      </ssa:IDT542_response>
   </soap:Body>
</soap:Envelope>
```

Delete Operation: Input Message

```
<?xml version="1.0" encoding="UTF-8"?>
    <soap:Envelope
                        xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
                        xmlns:real="http://www.identitysystems.com/xmlschema/iss-
version-8/RealTimeSync">
        <soap:Header/>
<soap:Body>
        <ns1:IDT003 xmlns:ns1="http://www.identitysystems.com/xmlschema/iss-version-8/
RealTimeSync" IDS_OP="A" IDS_SEQ="<ns1:AcctNo>144</ns1:AcctNo>
        <ns1:Address>9 TORRENS STREET</ns1:Address>
        <ns1:CL_ID/>
        <ns1:DOB>19661017</ns1:DOB>
        <ns1:Name>MARSHALL ROBERT</ns1:Name>
    </ns1:IDT003>
</soap:Body>
</soap:Envelope>
```

Delete Operation:Output Message

```
<?xml version="1.0" encoding="UTF-8"?>
        <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
                <soap:Header/>
                <soap:Body>
                <ssa:IDT003_response
xmlns:ssa="http://www.identitysystems.com/xmlschema/iss-version-8/RealTimeSync">
<ssa:response>0</ssa:response>
<ssa:status>1</ssa:status>
    <ssa:AuditMSG>
            <ssa:System>ssa003</ssa:System>
            <ssa:IDT>IDT003</ssa:IDT>
            <ssa:Rulebase>ids:rb</ssa:Rulebase>
            <ssa:ClusterAction>
        <ssa:Type>Delete from Cluster</ssa:Type>
            <ssa:To>
                    <ssa:ID>AA</ssa:ID>
                    <ssa:No>654</ssa:No>
            </ssa:To>
        <ssa:New>false</ssa:New>
```

```
        <ssa:Automatic>true</ssa:Automatic>
        </ssa:ClusterAction>
        </ssa:AuditMSG>
        </ssa:IDT003_response>
    </soap:Body>
</soap:Envelope>
```

Creating an Axis2 Java Proxy

Alternatively, you can create an Axis2 Java proxy with the supplied `compile.bat` script and the `Axis2Sample6` in the java-xml directory. You must have WS-Addressing and WS-Security (Rampart) options installed.

Custom HTTP Header

You can add one or more custom HTTP headers to a SOAP response. Use the custom HTTP headers to pass additional information such as authentication details or location of the Platform for Privacy Preferences (P3P) policy reference file.

To add a custom HTTP header to a SOAP response, perform the following tasks:

1. Create a configuration file in the following directory and name it as `ssahttp.xml`:
   - On Windows: `<Identity Resolution Installation Directory>\bin`
   - On UNIX: `<Identity Resolution Installation Directory>/bin`
2. Add the following entries to the `ssahttp.xml` file:

```
<httpProtocol>
   <customHeaders>
      <add name="<Header Name>" value="<Header Value>"/>
   </customHeaders>
</httpProtocol>
```

3. Replace the following entries with appropriate values:

   Header Name

   Name of the header

   Header Value

   Information that you want to add to the response.

   For example:

```
<httpProtocol>
   <customHeaders>
      <add name="X-Custom-Header" value="Passed"/>
   </customHeaders>
</httpProtocol>
```

4. Save the file and close it.

A running server rereads the configuration file by using the following flush command and adds the custom HTTP header to the response message:

- On Windows: `%SSABIN%\ssashut -h%SA_XMHOST% -f`
- On UNIX: `$SSABIN/ssashut -h$SA_XMHOST -f`

An example response is as follows:

```
HTTP/1.1 200 OK
Date: Fri, 31 Jan 2014 03:28:54 GMT
User-Agent: Informatica-IR/9.5.4
```

```
X-Custom-Header: Passed
Content-Length: 0
```

UDDI

IIR web services can be registered with UDDI.

Enabling UDDI registration

UDDI registration will not be carried out unless it has been enabled. This is done either though setting a number of environment variables, or with an XML configuration file.

UDDI Environment Variables

You must add these environment variables to `env\isss.bat` (Windows) or `env/isss` (UNIX).

Note: The environment variables are used by the Web Services servers, not the clients.

SSA_UDDI_DICT

The UDDI configuration file. If this is provided, its contents will take priority over the other environment variables.

SSA_UDDI_BUSINESS_NAME

The UDDI Business Name. The name of the party publishing the service.

SSA_UDDI_UID

The userid to log on to UDDI with.

SSA_UDDI_PWD

The password for logging on to the UDDI publisher.

SSA_UDDI_URL_INQUIRE

The UDDI inquiry URL, which is probably something like, `http://uddi/uddipublic/inquire.asmx`

SSA_UDDI_URL_PUBLISH

The UDDI publisher URL, which is probably something like, `http://uddi/uddipublic/publish.asmx`

UDDI Configuration file

The configuration file is identified by the environment variable `SSA_UDDI_DICT`. It is an XML file that contains cards for the UDDI environment. These cards have the same corresponding values as the similarly named environment variables.

The UDDI configuration file uses the namespace:`http://www.identitysystems.com/xmlschema/iss-version-8/uddi`

The configuration file is an XML file which looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This file was generated by:
Program: uddi.c
Version: $Change: 470146 $
Date : Wed Mar 03 13:45:41 2010
-->
<uddi xmlns="http://www.identitysystems.com/xmlschema/iss-version-8/uddi">
            <business>Informatica</business>
```

```
            <inquire>http://uddi/uddipublic/inquire.asmx</inquire>
            <publish>https://uddi/uddipublic/publish.asmx</publish>
            <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                                    Type="http://www.w3.org/2001/04/xmlenc#Element">
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
        <CipherData>
            <CipherValue>
MGjGOmXO01Hk9X2jlpHghcEoPg3/+UjRXVnqX0gvvx8Afc70mqtXCU2y1x3j/1HcyOb0he8KzpzY
BQIG/xFcqjRefOgOWlz7d3DWQfOBAmwoSCNXA44gjnM/rAAZVR2ztdDPlFdRR8kXEoAhBk2wXrIP
nAeRDRrqvqbnJj7t9v8liyLGTT+l2qaxt4GDzi6ysY4ag/G1lee7WkS6SQly6TLZr+5jVx2hdGsN
3ys+F80=</CipherValue>
        </CipherData>
</EncryptedData>
</uddi>
```

The values are:

business

The UDDI Business Name. The name of the party publishing the service.

inquire

The UDDI inquiry URL, which is probably something like:`http://uddi/uddipublic/inquire.asmx`

publish

The UDDI publisher URL, which is probably something like: `http://uddi/uddipublic/publish.asmx`

user

The UDDI user, which must be in the UDDI publishers group. This item is encrypted.

password

The UDDI user's password and this item is encrypted.

UDDI configuration tool

Because the UDDI configuration file contains encrypted elements, it is created using the UDDI configuration tool, which is called %SSABIN%\uddiconf.exe on Windows and $SSABIN/uddiconf on Unix. It prompts for the required items and creates the file.

```
uddiconf SSAIO 9.0.0.01 MSVS2005 Mar 3 2010 12:55:08 IDS9.0.01
file (d:\a1ni\bin\uddi.xml):
user (ssa):
password:
re-enter:
business: Informatica
inquire: http://uddi/uddipublic/inquire.asmx
publish: https://uddi/uddipublic/publish.asmx
```

UDDI and IIR concepts

IIR concepts are mapped onto UDDI ones.

Business Entity

The UDDI Business entity is that supplied by the user through the `SSA_UDDI_BUSINESS_NAME` environment variable.

Business Service

This will be "Search", "XML Console" or "Synchronizer" depending on the service.

tModel

This will be the IIR system name.

Binding

This will contain:

1. The access point, which is the URL of the web service server
2. The search name in the IIR system
3. The Rulebase name corresponding to the system
4. The overview document, which I the URL of the WSDL document.

From this, it should be possible to construct a search request.

Using UDDI to discover searches

The `uddi.cs` sample demonstrates the use of UDDI to discover IIR searches. It requires the `.NET` UDDI SDK to compile it.

CHAPTER 10

ASM Workbench

This chapter includes the following topics:

- Introduction, 161
- Launching the ASM Workbench, 161
- ASM Workbench Input Options, 162
- ASM Workbench and Batch Test utility, 174

Introduction

The ASM Developer's Workbench is a Java GUI tool that helps a programmer prototype Address Standardization API calls. The ASM Workbench is used to parse addresses into their individual component fields and to validate them against postal reference databases. The ASM Workbench is used for:

- Parsing Unfielded Address Format
- Validating Fielded and Unfielded Address Format

In order to use the ASM Developer's Workbench, Identity Resolution (IIR) core modules should have been installed, either locally, or on another computer/server.

Launching the ASM Workbench

ASM Workbench can be launched from Identity Resolution `%SSABIN%` directory.

Command line startup for ASM Workbench

To run ASM Workbench, at the command prompt, run `%SSABIN%\env\isss.bat` to establish the environment and start the servers. After you establish the environment and start the servers, run `asmcli` in the Identity Resolution client environment.

The following are the main input parameters for launching ASM Workbench:

```
asmcli -hHostName:PortNumber -1File1 -2File2 [options]
```

where

-hHostName:PortNumber

Search Server Host name and Port Number.

1File1

Specifies the file where log message are redirected.

2File2

Specifies the file where error message are redirected.

-b

Use ASM with AddressDoctor v5.

-cCharacterSet

The Character set to use. The default is `WIN1250`.

-dDefaultCountry

The Country to use when parsing can not determine a country from the address.

-mValidationMode

The Mode to use for validation purposes valid values are `Suggest`, `Correct`, `Complete` and `Certify`. The default value is `Suggest`.

For example:

```
%SSABIN%\asmcli -h%SSA_SEHOST% -1asmcli.log -2asmcli.err
```

or

```
%SSABIN%\asmcli -h%SSA_SEHOST% -1asmcli.log -2as0mcli.err -d""
```

ASM Workbench Input Options

Country Specific Input

Country specific Input:

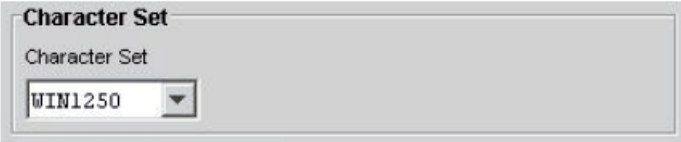
Force Country option

Force country option is used to force the use of default country.

Note: Before selecting a Country Name for validating an address or preloading country database into memory, appropriate Postal database (`.MD`) file must be present in `%SSATOP%/ssaas/ad/ad/db` directory for ASM using AddressDoctor v4 and in `%SSATOP%/ssaas/ad5/ad/db` directory for ASM using AddressDoctor v5.

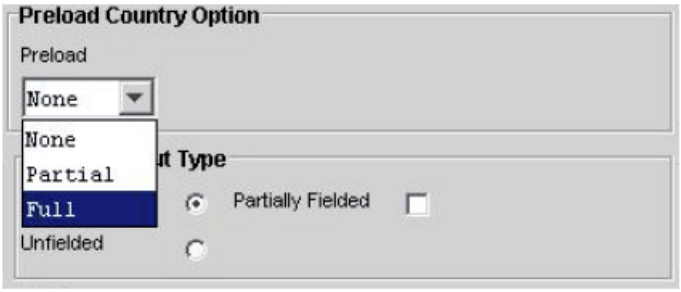
Character Set

This is used to define the character set of the input data. The parsed / validated address fields will be returned to the caller using this character set as well.



Country Preload Option

Preload option provides greater flexibility in loading the country address database into memory. The preload option includes partial preload for US CASS (certified) including ZIP move and EWS data. Only one country database can be preloaded. If database file is not located or insufficient memory, causes preload to fail.



Partial Preload

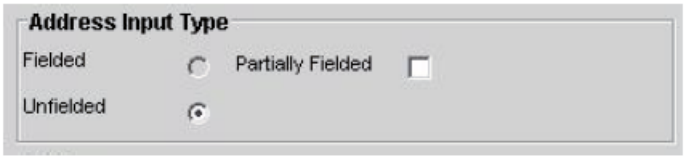
Partial preloading will load the data and indexing structures into memory. The reference data itself will remain on the hard drive. Partial is an alternative when not enough memory is available to fully load the desired databases.

Full Preload

Full preloading will move the entire reference database into memory. This may need a significant amount of memory for countries with large databases such as USA or, but it will increase the processing speed significantly.

Note: Before preload of country verify selected country name contains corresponding database (`.MD`) file located in appropriate postal reference database directory `%SSATOP%/ssaas/ad/ad/db` for ASM using AddressDoctor v4 and `%SSATOP%/ssaas/ad5/ad/db` for ASM using AddressDoctor v5.

Address Input Type



Fielded address Input

Fielded addresses will typically provide the most reliable results when cleansing an address. This address input provide separate field for each address component input.

Partially Fielded Input

In many databases address data has been partially broken out. for example a separate state, or postal code field. But some of the address is left in generic "address lines". In this case the address data are input using the fielded data (example: Contact, Province, Locality, Country, Postal Code) by selecting Fielded address input type, and then use the DeliveryAddressLines to input the `ADDRESS_LINE_*` elements, this is done by selecting Partially Fielded checkbox.

UnFielded Input

UnFielded Address Input has no explicit structure (other than 10 line input) this input is most flexible, but produce least reliable results.

Options

Batch Mode

Batch Mode is mainly used for importing input file containing unfielded input data and correcting the input address.

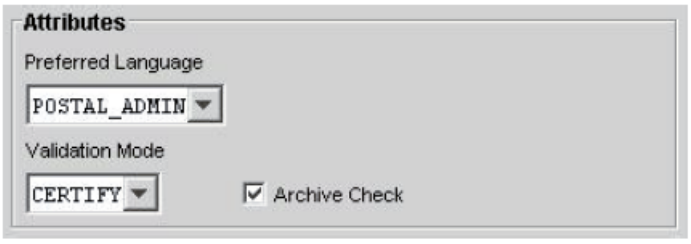
Interactive Mode

Interactive Mode is user driven, user has to use either unfielded (10 line input) or fielded inputs (individual components address input).

Parsing and Validation Frame

ASM Workbench provides four different operations for parsing, validating, certify input address and reset for clearing the input fields. For Fielded Address Format type Parse button will be disabled and for certify validation mode Validation button will be disabled.

Attributes



Preferred Language

This option is used to represent address into appropriate language type.

Option Value	Meaning
POSTAL_ADMIN	Set preferred language to that which is preferred by the postal service. This is the default.
LATIN_SCRIPT	Return the address using Latin script.
ENGLISH	English version of the address.

Validation Mode

Some optional aspects of Address Standardization behavior may be set by selecting the Validation Mode combo box.

Option Value	Meaning
Correct	Correct the input address. Do not generate suggestions. Generally used in batch mode.
Suggest	Generate suggestions (the default). Generally used by online applications where the operator can choose between a list of possibilities.
Complete	Use an incomplete address (fragment) to quickly generate suggestions. Used for online "fast completion" style of applications.
Certify	Use CASS certified validation rules defined by the USPS. Certify mode is only available for US addresses and requires additional database files to be installed in the DB directory.

Archive Check

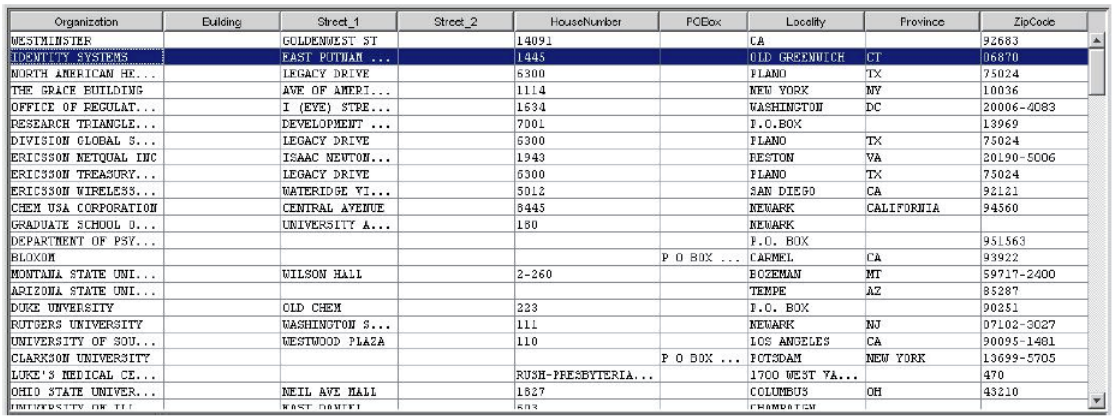
Archive Check option will include vanity names and outdated names (especially for localities) in the processing. Skipping this option will improve the speed very slightly, but may not correct addresses containing vanity names or outdate locality names. Two countries where you should definitely use this option are Germany and the US.

Suggested Address Label Display

The address shown in the suggested address label display is formatted according to the address formatting rules in the country

Address Result Panel Display

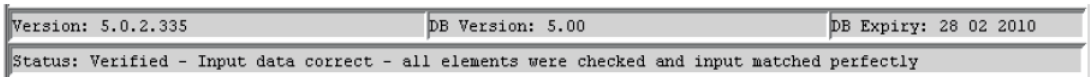
After parsing or validation, individual address fields are available for collection as part of a "suggestion". Suggestion 0 always holds the parsed fields. Suggestions numbered 1 and above hold the validated address fields. Individual fields are viewed in the List box in Output Results Frame. When the users has chosen the **UnFielded** and **validate** option, then Output frame list out the suggestion for the input address, once the user clicks on each suggestion the output address data from the List is mapped to corresponding fields in Fielded address display panel.



Organization	Building	Street_1	Street_2	HouseNumber	POBox	Locality	Province	ZipCode
WESTMINSTER		GOLDENWEST ST		14091		CA		92683
IDENTITY SYSTEMS		EAST PUTNAM ...		1445		OLD GREENWICH	CT	06870
NORTH AMERICAN HE...		LEGACY DRIVE		6300		PLANO	TX	75024
THE GRACE BUILDING		AVE OF AMERI...		1114		NEW YORK	NY	10036
OFFICE OF REGULAT...		I (EYE) STRE...		1634		WASHINGTON	DC	20006-4083
RESEARCH TRIANGLE...		DEVELOPMENT ...		7001		P.O.BOX		13969
DIVISION GLOBAL S...		LEGACY DRIVE		6300		PLANO	TX	75024
ERICSSON NETQUAL INC		ISAAC NEUTON...		1943		RESTON	VA	20190-5006
ERICSSON TREASURY...		LEGACY DRIVE		6300		PLANO	TX	75024
ERICSSON WIRELESS...		WATERIDGE VI...		5012		SAN DIEGO	CA	92121
CHEM USA CORPORATION		CENTRAL AVENUE		8445		NEWARK	CALIFORNIA	94560
GRADUATE SCHOOL O...		UNIVERSITY A...		180		NEWARK		
DEPARTMENT OF PSY...						P.O. BOX		951563
BLOXOM					P O BOX ...	CARMEL	CA	93922
MONTANA STATE UNI...		WILSON HALL		2-260		BOZEMAN	MT	59717-2400
ARIZONA STATE UNI...						TEMPE	AZ	85287
DUKE UNVERSITY		OLD CHEM		223		P.O. BOX		90251
RUTGERS UNIVERSITY		WASHINGTON S...		111		NEWARK	NJ	07102-3027
UNIVERSITY OF SOU...		WESTWOOD PLAZA		110		LOS ANGELES	CA	90095-1481
CLARKSON UNIVERSITY					P O BOX ...	POTSDAM	NEW YORK	13699-5705
LUKE'S MEDICAL CE...				RUSH-PRESBYTERIA...		1700 WEST VA...		470
OHIO STATE UNIVER...		NEIL AVE MALL		1827		COLUMBUS	OH	43210
UNIVERSITY OF IL...		WEST DANIEL		503		CHAMPAIGN		

Validation Status and Database Version Display

Status bar displays Validation postal database version, AddressDoctor version and the status message when we press the validation button to validate the address.



Where version represents the Validation Database version and the status represent the Validation Status as mention in the following table for ASM using AddressDoctor v4:

Status Code	Meaning
0	Address is correct
1	Address was corrected
2	Needs correction; deliverability high
3	Needs correction; deliverability fair
4	Needs correction; deliverability small
5	Country not recognized
6	No valid country database found
7	Country not unlocked

Status Code	Meaning
8	No validate called yet
9	Insufficient information
10	No suggestions
11	Suggestions incomplete
12	Suggestions

Refer below table for validation status for ASM using AddressDoctor v5:

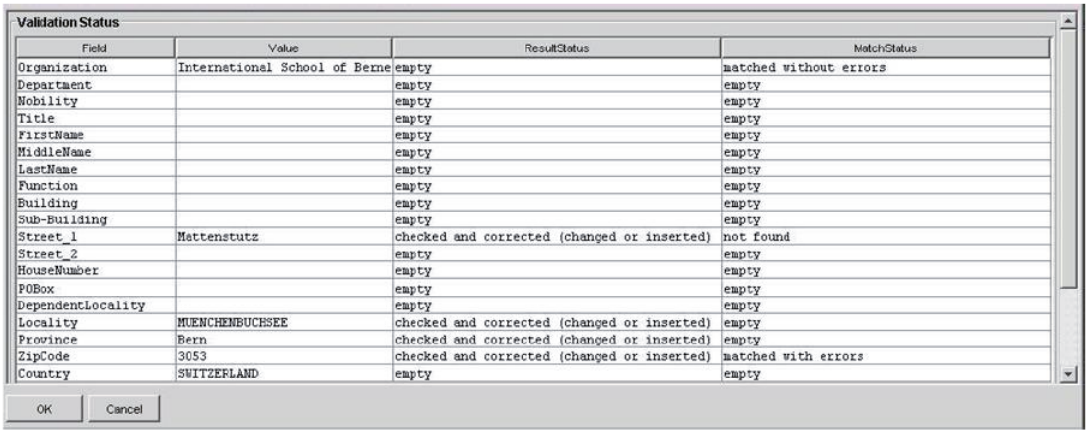
Status Code	Meaning
0	Verified - Input data correct - all elements were checked and input matched perfectly
1	Verified - Input data correct on input but some or all elements were standardised or input contains outdated names or exonyms
2	Verified - Input data correct but some elements could not be verified because of incomplete reference data
3	Verified - Input data correct but the user standardisation has deteriorated deliverability
4	Corrected - all elements have been checked
5	Corrected - but some elements could not be checked
6	Corrected - but delivery status unclear
7	Corrected - but delivery status unclear because user standardisation was wrong
8	Data could not be corrected completely, but is very likely to be deliverable - single match
9	Data could not be corrected completely, but is very likely to be deliverable - multiple matches
10	Data could not be corrected, but there is a slim chance that the address is deliverable
11	Data could not be corrected and is pretty unlikely to be delivered
12	FastCompletion Status - Suggestions are available - complete address
13	FastCompletion Status - Suggested address is complete but combined with elements from the input
14	FastCompletion Status - Suggested address is not complete
15	FastCompletion Status - Insufficient information provided to generate suggestions
16	Country recognized from ForceCountryISO3 Setting
17	Country recognized from DefaultCountryISO3 Setting

Status Code	Meaning
18	Country recognized from name without errors
19	Country recognized from name with errors
20	Country recognized from territory
21	Country recognized from province
22	Country recognized from major town
23	Country recognized from format
24	Country recognized from script
25	Country not recognized - multiple matches
26	Country not recognized
27	Parsed perfectly
28	Parsed with multiple results
29	Parsed with Errors - Elements change position
30	Parse Error - Input Format Mismatch
31	Validation Error: No validation performed because country was not recognized
32	Validation Error: No validation performed because required reference database is not available
33	Validation Error: No validation performed because country could not be unlocked
34	Validation Error: No validation performed because reference database is corrupt or in wrong format
35	Validation Error: No validation performed because reference database is too old - you need to contact AddressDoctor to obtain updated reference data

Output Result Frame Column Selection Menu

Output Result Frame also provides option to select the list of column that user wants to view on the Output result panel. When the user right clicks on the list panel then popup menu displays on the screen showing the list of columns that user wants to enable or disable on the list.

Field Status Display



Validation Status

Field	Value	ResultStatus	MatchStatus
Organization	International School of Berne	empty	matched without errors
Department		empty	empty
Nobility		empty	empty
Title		empty	empty
FirstName		empty	empty
MiddleName		empty	empty
LastName		empty	empty
Function		empty	empty
Building		empty	empty
Sub-Building		empty	empty
Street_1	Mattenstutz	checked and corrected (changed or inserted)	not found
Street_2		empty	empty
HouseNumber		empty	empty
POBox		empty	empty
DependentLocality		empty	empty
Locality	MUENCHENBUCHSEE	checked and corrected (changed or inserted)	empty
Province	Bern	checked and corrected (changed or inserted)	empty
ZipCode	3053	checked and corrected (changed or inserted)	matched with errors
Country	SWITZERLAND	empty	empty

OK Cancel

Match Status

All results share a Match Status that describes how the address elements matched to the postal reference data. Refer below table for match status for ASM using AddressDoctor v4:

Match Status	Meaning
0	Empty
1	Not found
2	Not Checked (no reference data or no chance of success)
3	Matched with errors
4	Matched without errors

Refer below table for match status for ASM using AddressDoctor v5:

val_status	Meaning
0	Empty
1	Not found
2	Not checked (no reference data)
3	Wrong - Set by validation only
4	Match with errors in this element
5	Match with changes
6	Match without errors

Result Status

The Result Status indicates for each address component if and how it has been modified during the address validation process. Refer below table for result status for ASM using AddressDoctor v4:

Result Status	Meaning
0	Empty
1	Not checked
2	Not checked but standardized
3	Checked and corrected (changed or inserted)
4	Validated, but changed (synonyms, old names)

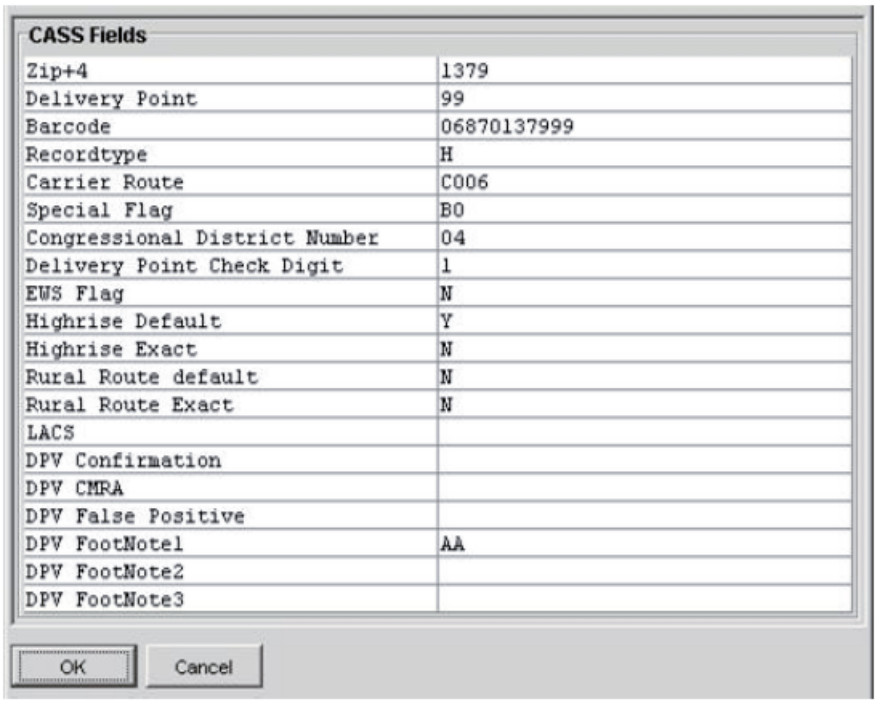
Result Status	Meaning
5	Validated, but standardized
6	Validated and unchanged

Refer below table for result status for ASM using AddressDoctor v5:

val_mods	Meaning
0	Empty
1	Not validated and not changed
2	Not validated bus standardized
3	Validated but not changed due to invalid input
4	Validated but not changed due to lack of reference data
5	Validated but not changed due to multiple matches
6	Validated and changed by eliminating the input value
7	Validated and changed due to correction based on reference data
8	Validated and changed by adding value based on reference data
9	Validated, not changed, but delivery status not clear
12	Validated, verified but changed due to outdated name
13	Validated, verified but changed from exonym to official name
14	Validated, verified but changed due to standardization based on casing or language
15	Validated, verified and not changed due to perfect match

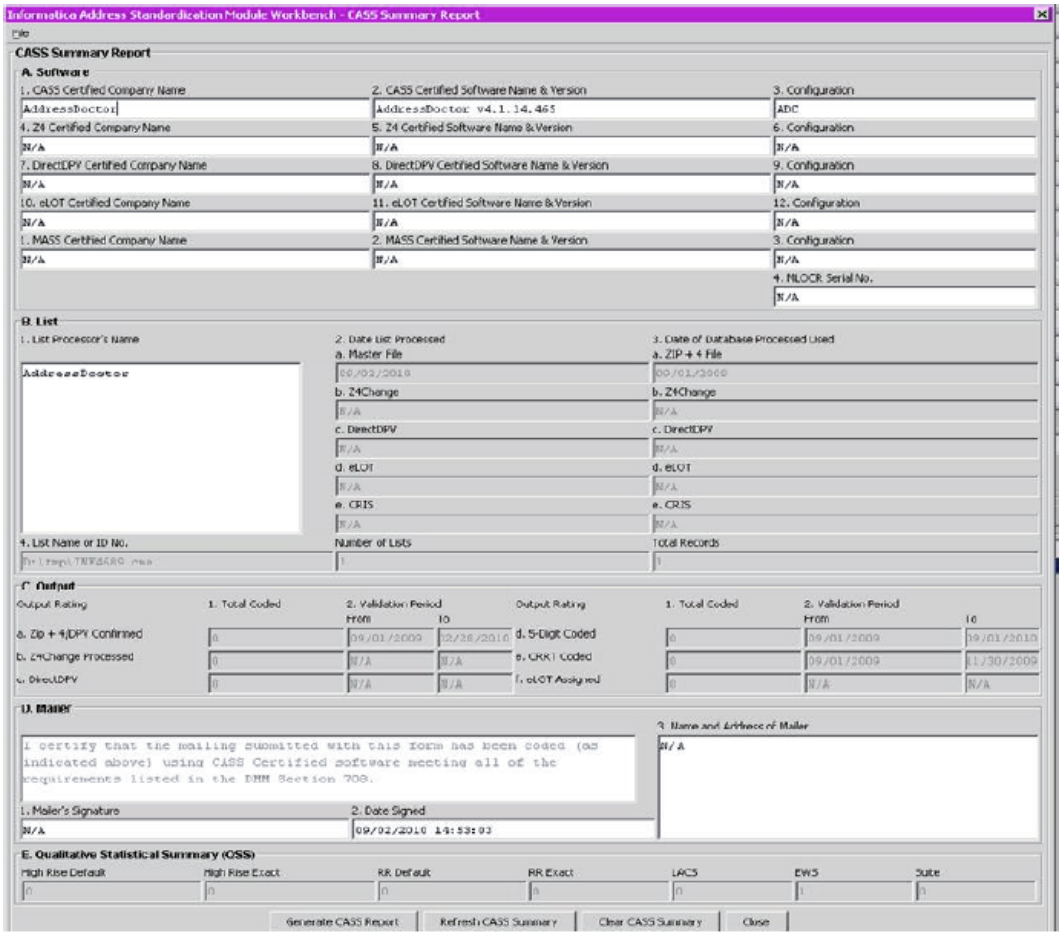
CASS Field Status Display

When validating address using validation mode as Certify, The Result panels shows validated address fields and when Display CASS Fields is selected will pop up CASS fields dialog as shown below:



CASS Summary Report Display

USPS Form 3553 CASS certification can be generated by selecting **View CASS Report** from popup menu displayed in the address result tab for CASS certified address. Selecting this option displays **CASS Summary** report dialog as shown below:



Statistics Reports - CASS Certification

ASM also provides options to generate USPS CASS 3553 Summary report which displays total records coded in each category. The report can be generated in HTML and XML format. CASS 3553 summary report sections are explained in detail under **CASS Summary Report Display**.

File Menu Options

File | Clear Results

Menu option **File** > **Clear Results** menu option is used to clear the address result output display windows.

File | Save Input

Menu option **File** > **Save Input** will prompt File Dialog, this options reads the file name from user and dumps address input and options(including validation mode, preload option, attributes) in the input file, which can be used as input for ASM Batch Test utility.

File | View CASS Summary

Menu option **File** > **View CASS Summary** will display CASS Summary report dialog.

File | Generate CASS Report

Menu option **File** > **Generate CASS Report** will prompt File Dialog, this options reads the file name from user and generate CASS summary for the input address. File format can be either XML or HTML.

File | Clear CASS Summary

Menu option **File** > **Clear CASS Summary** is used to clear accumulated CASS summary after multiple validation using validation mode as certify.

File | Exit

Menu option **File** > **Exit** will prompt to close and exit all SSA-NAME3 Workbench sessions.

ASM Workbench and Batch Test utility

Using ASM Workbench, select **File** > **Save Input**option, the current Fielded or Unfielded Workbench input can be dumped to flat file, this flat file can be used as input for the ASM Batch Test utility `asmiss`.

This is a sample dump of UnFielded Address Input created by the ASM Workbench:

```
# ***Informatica's ASM input file***
# -hlocalhost:1666
# -dSwitzerland -cWIN1250 -l -mSuggest -v -yENGLISH -z -S -a -A -th
WOLFTRONIC Disco & Concert Equip.
zur Brunnenstube
Aeugst am Albis
CHE
```

This is a sample dump of Fielded Address Input created by the ASM Workbench:

```
# ***Informatica's ASM input file***
# -hlocalhost:1666
# -dSwitzerland -cWIN1250 -i -l -mSuggest -v -yENGLISH -z -S -a -A -th
00 International School of Berne
01
02
03
04
05
06
07
08
09
10 Mattenstutz
11
12
13
14
15 MUENCHENBUCHSEE
16 Bern
17 3053
18 Switzerland
#
```

The command to run the ASM Batch Test utility is as follows:

```
SSABIN%\asmiss [-h<host:port>] <ASM Workbench generated input file>
```

Note: The -h option must be specified either in the input file or on the command line. The command line overwrites any value specified in the input file.

CHAPTER 11

System Backup and Restore

This chapter includes the following topics:

- Overview, 176
- Back Up the System, 176
- Restore the System, 177

Overview

Use the operating system and database backup tools to back up the Identity Resolution systems, and the tools can vary from system to system.

The first step is to ensure that the Identity Resolution databases and Rulebase are in a stable state. The best way to do that is to shut down the Identity Resolution Rulebase and connection/search servers. Use the Identity Resolution Console or run the `idsdown` script to shut down the Identity Resolution Rulebase and connection/search servers. On UNIX, use the `$SSABIN/idsdown` command.

After you shut down the Identity Resolution servers, use the database vendor tools to back up all the Identity Resolution tables and databases. For example, with IBM DB2 UDB, you can use the `BACKUP DATABASE` command. Oracle has multiple ways to back up databases. For example, the Oracle Recovery Manager. Save the user information. IBM DB2 UDB uses system user IDs. A complete system backup is the best way to ensure that all the information is backed up. If a complete backup is not feasible, document all the scripts and procedures that you used to set up the Identity Resolution system so that a complete restore is possible.

Back Up the System

1. Shut down the Identity Resolution server processes to ensure that all the changes are written to databases.
2. Use vendor database tools to back up the Identity Resolution databases. Also, back up users and user permissions if they are not backed up by this step. Note that IBM DB2 UDB uses system user IDs.
3. Use the operating system backup tools to back up the Identity Resolution directories.

Restore the System

1. Restore the Identity Resolution directories. Follow the Identity Resolution install procedures for setting the system path, shared library path, and environment variables.
2. Restore user information (re-create user ID and permissions).
3. Restore databases.
4. Start the Identity Resolution servers.

It is a good practice to have a standard test or a set of tests to verify the restore.

CHAPTER 12

Batch Utilities

This chapter includes the following topics:

- Batch Utilities, 178
- Common Parameters, 179
- ssachdb Utility, 180
- dbinit Utility, 181
- idsinit Utility, 181
- lockmgr Utility, 181
- version Utility, 182
- idsbatch Utility, 182
- iirconfig-tool Utility, 183
- loggrabr Utility, 185
- logfrmat Utility, 186
- db_util Utility, 187
- Command File Syntax, 187

Batch Utilities

Use the batch utilities to run the Identity Resolution processes in the command line.

The following table lists the batch utilities that you can run in the command line:

Function	Utility Name
Creating a Rulebase	idsinit, idsbatch
Creating a database	dbinit, idsbatch
Creating a system	idsbatch
Deleting a system	idsbatch
Changing the database connection string	ssachdb

Function	Utility Name
Version signatures	version
Running a batch job	idsbatch
Stopping a batch job	idsbatch (if you use idsbatch to start the job)
Listing or removing locks	lockmgr, db_util
Table Loader	idsbatch

Common Parameters

When you run a batch utility, you must specify the required input parameters in the command line.

Most of the batch utilities use the following parameters:

-pSystem

The System Name.

-rRbName

Rulebase name.

-hRbHost

Rulebase Server connection details. Format is `hostname:port`.

-dDbName

Database name.

-vVerbosity

Verbosity of job where p=progress, s=statistics, u=usage, i=info.

-1UtilityLog

Utility Log file name.

-2UtilityErr

Utility Error file name.

--1ServerLog

Server Log file name.

--2ServerErr

Server Error file name.

ssachdb Utility

This utility is used to change the source or target database connection string, stored in the Rulebase, for a given system. The utility may also be used to change the schema name associated with the source database (defined in the User Source Tables section).

This is useful when the DBMS password has been changed, or when the entire Rulebase has been copied and moved to another database (example, Test to QA). The syntax is:

```
%SSABIN%\ssachdb -oOldName -nNewName [-z] | -xOldSchema -yNewSchema CommonParms
```

where

-oOldName

The old Database connection string

-nNewName

The new Database connection string

-z

Update IDS_2PC table (see Synchronizer Considerations below).

-xOldSchema

The old source database schema

-yNewSchema

The new source database schema

For example, the following three calls change the name of the source connection string, source schema from TEST to QA, and target database connection string respectively.

```
ssachdb -oodb:99:ssa/ssa@srcdb -nodb:99:ssa/ssa@newsrcdb ...
ssachdb -xTEST -yQA ...
ssachdb -oodb:1:ssa/ssa@tgtdb -nodb:1:ssa/ssa@newtgtdb ...
```

The utility updates the Rulebase by performing a simple text replacement of the old string with the new one. If it is necessary to change the source and target connection strings independently, it is necessary to use different names for them in the original system. For example, using a database number of 99 for the source database distinguishes it from the target database, which by default uses a database number of 1, even if they share the same connection parameters (uid/pwd@svc).

Synchronization Considerations

In general, synchronization of a cloned target database is not possible without reloading the IDT/IDXs. There are many problems including:

- Triggers on the source database will continue creating transactions for a specific database number (example, IDS_01). If the new target database's connection string specifies a different database number, it will be unable to see those transactions.
- There are now multiple consumers of trigger transactions (databases A and A'), but only one transaction will be created, which is insufficient to synchronize both databases. In other words, the cloned system does not have any triggers defined for it, and must not process the transactions created for the original system.
- The `IDS_2PC` table contains the name (connection string) of the original target database. This may be changed to the new target database by using the -z switch in combination with the -o and -n switches. That is, `ssachdb` will connect to the database specified with-n and update the `IDS_2PC` table, replacing the `OldName` with the `NewName`.

dbinit Utility

This utility initializes a database ready for use by Identity Resolution.

```
%SSABIN%\dbinit [-pSystem] [-rRbName] [-hRbHost] -dDbName [-vpsu] [-f] [-gn]
```

where

-f

Deletes the database initialization flag but does not delete the database contents (IDTs, IDXs, etc).

-gn

n is the database granularity. This must be a power of two, and may be expressed in k or m (example, -g32k)

All parameters are optional, except for the database name, but if a system is supplied, a Rulebase must also be supplied, and vice versa.

idsinit Utility

This utility initializes a Rulebase.

```
%SSABIN%\idsinit -rRbName -hRbHost [-vpsu] [-f]
```

where

-f

Deletes the Rulebase.

lockmgr Utility

This utility performs the following tasks:

- Deletes a group of locks that have a common unique identifier (`uq`).
- Lists all the application locks for a system in ascending order by `uq`.
- Lists the current Rulebase host in a Rulebase Server Group.

The lockmgr utility uses the following syntax:

```
%SSABIN%\lockmgr [list|del uq] -rRbName -hRbHost [-vpsu] [-l]
```

You must specify either the `list` or `del uq` option. Use the `list` option to list the application locks and their unique identifier values or the current Rulebase host. Use the `del uq` option to delete a group of locks that have a common unique identifier.

You can use the Rulebase Server connection details by using the `-h` option or the Rulebase Server Group name by using the `-g` option. You can use the `-l` option in conjunction with the `-g` option to list the Rulebase Server connection details in the `host:port` format.

The following example lists all the application locks for a system:

```
%SSABIN%\lockmgr list -rodb:0:userid/password@dbserver -hlocalhost:1668

    sorted locks:
    uq: updsync,1,1:1370482219:6150
      obj: idtsync
        id: system=mdm_customer idt=
          type: w
          lock: ip=127.0.33.1 pid=4560 host='rb.informatica.com' ps='2013/06/06
    01:30:18.2132'
    uq: updsync,1,1:1370482219:6150
      obj: idtsyncsystem
        id: system=mdm_customer
          type: r
          lock: ip=127.0.0.1 pid=4560 host='rb.informatica.com' ps='2013/06/06
    01:30:18.2132'
```

The following example deletes a group of locks based on the specified common unique identifier:

```
%SSABIN%\lockmgr del updsync,1,1:1370482219:6150 -rodb:0:userid/password@dbserver -
hlocalhost:1668
```

You must use this utility with caution. Removing the locks from a running process, such as Update Synchronizer, that requires exclusive use of certain system resources might lead to data corruption and errors.

If a process detects a conflicting lock, it tries to find out the status of the job that created the lock. It deletes the locks and creates its own if the original process has crashed.

In some instances, the new process might not be able find out the status of the original process. For example, if the original process runs on a different computer or is unresponsive. The `lockmgr` utility is useful in such scenarios to remove the lock manually.

version Utility

You can use this script to display the version signatures from the programs and objects in Identity Resolution.

```
%SSABIN%\version [product-id]
```

where

`product-id` is the name of the sub-system to display. Valid values include `all`, `ce`, `db`, `dl`, `lm`, `io`, `ut` and `cs`.

idsbatch Utility

This program reads and executes actions defined in a text file. Most commonly performed Console Client functions are available by running user defined jobs. See the *User-Job-Definition* section in the *DESIGN GUIDE* .

```
%SSABIN%\idsbatch -h<CShost> -i<commandFile> -1<logFile> [-2<errorFile>] [-3<dbgFile>]
```

where:

-h<CShost>

Required. Console Server `host:port` address.

-i<commandFile>

Required. The name of the file that contains the list of actions to be performed.

-1<logFile>

Required. The name of the output (log) file.

-2<errorFile>

Optional. The name of the error output file. This file will contain error messages if one or more steps fail.

-3<dbgFile>

Optional. The name of the debug output file that can contain error messages and additional information about any failures.

iirconfig-tool Utility

Use the iirconfig-tool utility to retrieve or update the configuration metadata of a system. The configuration metadata includes details about the system, identify table, identity index, identity fields, search configuration, and other system-related configuration.

The iirconfig-tool utility uses the following format:

```
$SSABIN/iirconfig-tool -p|--print [<Key>] -u|--update [<Key>=<Value>|<File Name>] -h|--
host <Connection Details of the Rulebase Server> -r|--rulebase <Name of the Rulebase> -
s|--system <Name of the Sysem> [-a|--audit-log [<File Name for Audit Log>]] [-t|--test]
[-e|--escape] [-f|--force] [-q|--quiet] [-v|--verbose]
```

The iirconfig-tool utility uses the following parameters:

-p|--print [<Key>]

Retrieves the configuration metadata of the specified key or all the keys of the specified system. If you do not specify any key, the utility retrieves the configuration metadata of all the keys.

For example, the following command retrieves the Compress_Method value of the (System)SalesSystem key:

```
$SSABIN/iirconfig-tool -h OrgHost:8080 -r odb:0:myuid/mypassw@oraserv -s SalesSystem
-p '(System)SalesSystem:Compress_Method'
```

Note: Retrieve the configuration metadata of all the keys at least once to view the format of the keys and their values.

-u|--update [<Key>=<Value>|<File Name>]

Updates the value of the specified key or the values of all the keys that you add to the specified input file. If you want to update multiple key values, add all the keys and their values to an input text file and specify the input file name as a parameter.

For example, the following command updates the Compress_Method value of the (System)SalesSystem key to Default:

```
$SSABIN/iirconfig-tool --host OrgHost:8080 --rulebase odb:0:myuid/mypassw@oraserv --
system SalesSystem --update '(System)SalesSystem:Compress_Method=[Default]'
```

The following command updates the values of all the keys that you add to the `update.txt` file:

```
$SSABIN/iirconfig-tool --host OrgHost:8080 --rulebase odb:0:myuid/mypassw@oraserv --
system SalesSystem --update update.txt
```

Note: Retrieve the configuration metadata of all the keys to view the format of the keys and their values. In the input file, you can specify the keys and their values in the same format.

-h|--host <Connection Details of the Rulebase Server>

Connection details of the Rulebase server.

Use the following format: `<Host Name of Rulebase Server>:<Port Number>`

For example, `-h OrgHost:8080`

-r|--rulebase <Name of the Rulebase>

Name of the Rulebase or the dictionary alias of the Rulebase name.

Use the following format for a Rulebase name: `<DB Type>:<Number>:<User>/<Password>@<Service>`

For example, `-r odb:0:myuid/mypassw@oraserv`

Use the following format for the dictionary alias of a Rulebase name: `ids:<Rulebase Alias>`

For example, `ids:rb`

-s|--system <Name of the System>

Name of the system. The utility retrieves or updates the metadata information of the specified system.

For example, `-s SalesSystem`

-a|--audit-log [<File Name for Audit Log>]

Optional. Name and directory path for the audit log file. The utility adds entries to the log file when you update any key values. By default, the utility creates the `iirconfig-tool-audit.log` file in the current working directory.

-t|--test

Optional. Runs the utility in the test mode and validates the keys and the values that you want to update. If the validation fails, you get a message about the failure. After you successfully validate the keys and the values, you can update them in the system.

For example, the following command validates the format of the (System)SalesSystem:Compress_Method key and its value:

```
$SSABIN/iirconfig-tool -h OrgHost:8080 -r odb:0:myuid/mypassw@oraserv -s SalesSystem
-u '(System)SalesSystem:Compress_Method=[From IDT]' -t
```

-e|--escape

Optional. Replaces the special characters with their respective URL-encoded characters when you retrieve the configuration metadata of the specified key or all the keys of the specified system.

For example, the following command replaces the special characters with their respective URL-encoded characters in the retrieved metadata:

```
$SSABIN/iirconfig-tool -h OrgHost:8080 -r odb:0:myuid/mypassw@oraserv -s SalesSystem
-p -e
```

-f|--force

Optional. Forces the utility to update the values of the specified keys without reloading the system even though the update requires a system reload.

-q|--quiet

Optional. Does not include the blank and non-editable keys in the output when you retrieve the configuration metadata of a system.

-v|--verbose

Optional. Displays the values of the updated keys after the utility successfully updates the key values.

For example, the following command updates the Comment parameter of the (System)SalesSystem key:

```
$SSABIN/iirconfig-tool -h OrgHost:8080 -r odb:0:myuid/mypassw@oraserv -s SalesSystem
-u '(System)SalesSystem:Comment=New Value' -v
```

If the utility successfully updates the (System)SalesSystem key, you get the following output:

```
(System)testx538:Comment old="Outdated Value" new="New Value"
```

loggrabr Utility

Use the loggrabr utility to retrieve the log entries from log files, error files, environment scripts, and configuration files and load the entires into a single output file. You can use the output file to troubleshoot any issues without having to refer multiple files.

The loggrabr utility uses the following format:

```
$SSABIN/loggrabr <Output File Name> [-f] [<Start Date>] [<Start Time>] [<End Date>]
[<End Time>]
```

The loggrabr utility uses the following parameters:

Output File Name

Name, extension, and directory path for the output file. By default, the utility creates the output file in the current working directory.

-f

Optional. Forces the utility to overwrite any existing file that matches the output file name.

Start Date and Start Time

Optional. Date and time from which you want to retrieve the log entries based on the time stamp. Use the `yyyy-mm-dd` format to specify the date and the `hh:mm` format to specify the time.

By default, the utility retrieves all the log entries in the current environment.

End Date and End Time

Optional. Date and time up to which you want to retrieve the log entries based on the time stamp. Use the `yyyy-mm-dd` format to specify the date and the `hh:mm` format to specify the time.

By default, the utility retrieves all the log entries in the current environment.

For example, the following command retrieves the log entries from multiple files and loads them into the `output.log` file:

```
$SSABIN/loggrabr output.log -f 2015-04-22 11:20 2015-03-31 21:30
```

After you run the loggrabr utility, you can use the logfrmat utility to format the log entries in the output file and improve the readability of the output file.

logfrmat Utility

Use the logfrmat utility to format the log entries in an output file of the loggrabr utility and load the formatted entries into an XML output file. You can use the logfrmat utility to improve the readability of its input file, which is an output file of the loggrabr utility.

The logfrmat utility groups the log entries of each log file listed in its input file into different sections, such as error summary, stack trace, and results. An error summary section describes the error, a stack trace section lists additional information about the error, and a result section lists the completed processes and their results.

The logfrmat utility also creates separate sections for the log entries retrieved from environment scripts and configuration files, SQL logs, and process logs in the output file.

The logfrmat utility uses the following format:

```
$SSABIN/logfrmat <Input File Name> <Output XML File Name> [-f] [<Start Date>] [<Start Time>] [<End Date>] [<End Time>]
```

The logfrmat utility uses the following parameters:

Input File Name

Name, extension, and directory path of the output file that the loggrabr utility creates. Default directory path is the current working directory.

Output XML File Name

Name and directory path for the output XML file. By default, the utility creates the output file in the current working directory.

-f

Optional. Forces the utility to overwrite any existing file that matches the output file name.

Start Date and Start Time

Optional. Date and time from which you want to format the log entries based on the time stamp. Use the `yyyy-mm-dd` format to specify the date and the `hh:mm` format to specify the time.

By default, the utility formats all the log entries in the output file of the loggrabr utility.

End Date and End Time

Optional. Date and time up to which you want to group the log entries based on the time stamp. Use the `yyyy-mm-dd` format to specify the date and the `hh:mm` format to specify the time.

By default, the utility formats all the log entries in the output file of the loggrabr utility.

For example, the following command formats the log entries in the `output.log` file and loads the formatted log entires into the `output.xml` file:

```
$SSABIN/logfrmat output.log output.xml -f 2015-04-22 11:20 2015-03-31 21:30
```

db_util Utility

The db_util utility unlocks a Rulebase Server after the server shuts down unexpectedly. When you run the db_util utility, the utility internally runs an SQL script that deletes the `IDS_nn_INUSE` table and removes the lock related entry in the `IDS_FDT_META` table.

To run the db_util utility, use the following command format:

```
db_util ids:rb [Rulebase Number] [-c<Utility Name>] [-s<SQL Script Name>]
```

The db_util utility uses the following parameters:

Rulebase Number

Optional. Rulebase number for the Rulebase Server that you want to unlock. By default, the utility uses the rulebase number that you specified in the connection string.

Utility Name

Optional. Utility to run the SQL script. Default is sqlplus.

SQL Script Name

Optional. Absolute path and name for the SQL script that the utility internally creates. By default, the utility creates an SQL script named, `myscript.sql`, in the current working directory. Use this parameter when you do not have the write permission to the current working directory.

The following sample command overrides the rulebase number in the connection string and unlocks the Rulebase Server:

```
$SSABIN/db_util ids:rb 03 -csqlplus -s/tmp/dbscript.sql
```

Command File Syntax

```
# <comment lines start with the '#' character>
ACTION=<action name>
<parameter>=["]parameter value["]
```

The keywords, such as ACTION are not case-sensitive. White space after the equal sign is optional as are quotes around parameter values. Parameter values can contain embedded environment variables that are evaluated at run-time. Environment variable names must be surrounded by '#' characters, for example `#SSAWORKDIR#` is a valid environment variable. Any empty lines are ignored.

At the beginning of the input file, a mandatory parameter work-directory= should be initialized with the full path of the desired Identity Resolution Server working directory. For example:

```
work-directory=c:\InformaticaIR\work
```

or assuming that the Identity Resolution Server has the *SSAWORKDIR* environment variable set (as it should):`work-directory=#SSAWORKDIR#`

The Identity Resolution Server working directory can be overridden for each individual action by giving the work-directory= parameter after the `action=` statement.

The following actions and parameters are supported.

Rulebase creation

```
action=rulebase-create
rulebase-name= <dbtype>:<number>:<user>/<password>@<service>
```

For example:

```
action=rulebase-create
rulebase-name="odb:0:myuid/mypassw@oraserve"
```

or using dictionary alias

```
action=rulebase-create
rulebase-name=ids:rulebase
```

Rulebase deletion

```
action=rulebase-delete
rulebase-name= <dbtype>:<number>:<user>/<password>@<service>
```

For example:

```
action=rulebase-delete
rulebase-name="odb:0:myuid/mypassw@oraserve"
```

Database creation

```
action=database-create
database-name=<dbtype>:<number>:<user>/<password>@<service>
```

For example:

```
action=database-create
database-name="odb:1:myuid/mypassw@oraserve"
```

or using dictionary alias

```
action=database-create
database-name=ids:database
```

System creation

```
action=system-create
system-name=<name of the system to be created>
sdf-name= <name of the system definition file which describes the new system>
rulebase-name=<dbtype>:<number>:<user>/<password>@<service>
database-name=<dbtype>:<number>:<user>/<password>@<service>
```

For example:

```
action=system-create
system-name=mysystem
sdf-name="#SSAWORKDIR#/mailinglist.sdf"
rulebase-name="odb:0:myuid/mypassw@oraserve"
database-name="odb:1:myuid/mypassw@oraserve"
```

System deletion

```
action=system-delete
system-name=<name of the system to be deleted>
rulebase-name=<dbtype>:<number>:<user>/<password>@<service>
```

For example:

```
action=system-delete
system-name=mysystem
rulebase-name="odb:0:myuid/mypassw@oraserve"
```

Run User-Defined Job (User Job)

```
action=job-run
job-name=<name of the pre-defined User Job to be run>
system-name=<name of the system to be deleted>
rulebase-name=<dbtype>:<number>:<user>/<password>@<service>
work-directory=<working directory of the Console Server>
```

For example:

```
action=job-run
job-name="run-name-relates"
system-name=mysystem
rulebase-name="odb:0:myuid/mypassw@oraserve"
work-directory=c:\InformaticaIR\work
```

After the job starts, a detailed message including the run number for each step is written to the output file.

Stop Job

```
action=job-stop
rulebase-name=<dbtype>:<number>:<user>/<password>@<service>
system-name=<name of the system associated with the job>
run-number=<number of a started job>
```

For example:

```
action=job-stop
system-name=mysystem
rulebase-name="odb:0:myuid/mypassw@oraserve"
run-number=1
```

When stopping a job started using `idsbatch`, check the output from the preceding `job-run` action to determine the value of the run-number parameter.

INDEX

-vVERBOSITY 45
.NET Proxy 152
%SSABIN% 161

A

Actions Set 115
Actions Sets 116
Address Input 162
AddressDoctor 162
Administrator search clients 61
Apache Axis2 122
Application Program Interface 20
Archive Check 162
ASM Batch Test
  SSABIN% 174
ASM Workbench 161, 174

B

BACKUP DATABASE 176
batch 178
batch client 108
Batch Mode 162
Batch Search Client 74
Batch Search Clients 59
Batch Searches 108
batch utility
  logfrmat 186
Business Component 115
Business Service 112

C

CASS Certification 162
CASS Summary 162
checkpoints 77
CJK characters 102
Clear Messages 48
Client INI 61
Client Selection INI file 61
Client Work Directory 47
Clone 57
Clustering Viewer 55
Command File 178
Command line 161
commit rate 99
Compress-Key-Data 79, 99
Configure Mode 24, 27
Connection Aliases 30
Connection server 66
Connection Server 20, 47, 60
Console client 45
Console Client 22, 45, 66
Console server 45
Console Server 22, 27, 48, 66
Control Objects 16
Control Record 96
Country Preload Option
  Partial Preload
    Full Preload 162
Country Specific
  AddressDoctor 162
Create Rulebase 178
Custom mode 150

D

database character set 103
Database Level 103
Database Management Systems 12
Database Object Names 12
dbinit 178
DBMS load utility 75
DBMS Load utility 80
DBMS loader 109
DEDUP-PROGRESS 73
define_source 71
Dictionary Alias 64, 65, 150
DupFinder 74
DupFinder function 73
DupFinder report 71

E

EAI Siebel Adapter 117
End-of-File marker 105
environment variables 38, 118
error logs 18

F

flat file 96
Flat-File 78, 83, 106

G

Generic Mode 147
Global Jobs 57
Global Logs 19

H

HTTP Search Server 64, 65

HTTP Server 59

I

Identity Indexes 75
ids_conv 109
ids_error_get 18
ids_search_dedupe_start 73
idsbatch 178
idsclie.ini 61
IDT layout 72
IDT Name 151
IDX entry 99
IIR Connection Server 27
IIR Utility Service 113
iirconfig-tool 183
Import System 57
Infile 66
INI file 63
Input Locale 102
Input Queue 78
integrity 99
Interactive Mode 162
Interface 12
ISSErrorHandler 113
ISSLaunchBuildLoadFile 118
isss.bat 38
ISSSYNC 116

J

Java Applet 60
Jobs menu 56

K

Key Generation 79

L

Launched Jobs 48
License Server 20
Lite client 61
Lite-Indexes 22
Live Progress 47
load process 112
Load-IDT 57
Loader threads 79
Loader- Definition 106
Loader-Definition 75
loadit 75
lockmgr 19
Log Viewer 57
loggrabr
  batch utility 185
Logical-File-Definition 66, 118

M

manual restart 30
Menu
  Database 50
  Rulebase 50
Menu *(continued)*
  Servers 50
  System 50
Microsoft SQL Server 103
Mode
  Configure 47
  Custom 64
  Generic 64
  Normal 47
MSQ 81
Multi-byte 105
Multi-Search Definition 73, 74
multi-threaded mode 71

N

NLS_CHARACTERSET 103
NLS_LANG 103
No Source Access
  NSA Transaction Table 83
Notification Service 149
NSA Transaction Table 20, 96, 119
NSA-Batch Service 20, 147

O

Object Manager 111
ODBC interface 12
Online Search Clients 59
Online Searches 108
Optimizer's statistics 99
Optional Switches 86
Oracle Client 103
Outfile 66

P

Parsing 162
Pre Delete Event 116
PreDeleteRecord 117
primary keys 94
Profile Attributes 115

R

RB Server 32
Real Time clients 149
Real Time Synchronization 151
Real Time Web Service 90, 148, 150, 152
Regression Test 24
Reject_Duplicate_PK 94
Rejected Time Stamp 151
relate 66
Relate 66
Relate Client 45
Repository Workflow 113
restart 94
Rulebase 48, 57, 178
RuleBase access 20
RuleBase Objects 16
Rulebase server 30
Rulebase Server 20, 24
Rulebase Server Groups 32
Rulebase Server Options 25

Run Tests 24
Run Time Events 117
Run-Time Events 120

S

Sample Server Start-up 29
Script Coding 42
SDF 118
SDF File 50, 152
Search Client 20, 61
Search Server 18, 66
Search Server Host 161
Search-Definitions 59
Server Shutdown 29
Server-side Search 108
Service Control Manager 42
Siebel Business Component 112
Siebel Connector 20
Siebel CRM 20
Siebel Integration Object 112
Siebel Restrictions 120
SOAP 121
Soap messages
  examples 153
SOAP standard 121
Sort utility 78
SQL*Loader 81, 83
SSA Program 45
SSA_DBDICT 12
SSA_HTPORT 64
SSA_SERVER_STATS 31
SSA_XML_SIZE 118
SSA_XSHOST 148
ssashut 29
ssasrsv 22
SSATEMP 38
SSAWORKDIR 61
Start Script 42
Step Logs 19
Switches 25
synchronization 119
Synchronization 178
Synchronization Level 94
Synchronization Server 20, 147
Synchronizer 94
Synchronizer Objects
  System Loader 16
  Update Synchronizer 16
Synchronizer Utilities 83
System Authentication 12
system backup 176
System Editor 57
System Jobs 57
system restore 176
System-Qualifier 16
SystemQualifier 12

T

Table Loader 75, 77, 106, 119
Tablespace 94
Target Column Size 106
TCP/IP sockets 20
Transaction Record 96
Transport Layer Security 122
triplet 72

U

UDB database 79
UDDI 158
UDDI Configuration file 158
UDDI Environment Variables 158
Unfielded Address 161
Unicode 121
UNICODE 102, 106
Update Synchronizer 19, 80, 83, 86
updmulti 90
updsync 86
User Source Tables 99, 109
UST 101, 112

V

Verbosity 29

W

web services 121
Windows Service 29
Windows Services 42
Workflow 112
Workflows 113
working directory 178
WSDL file 132, 147
WSDL files 152

X

XML Console Server 20
XML Console Service 132
XML file 71
XML Search Service 122
XS Server 111, 119, 120
XSLT clause 71
XSLT stylesheet 71