



AGETOR®

ABF Application
Integration Guide

Content

1	Preface.....	3
1.1	Audience	3
1.2	Typographic conventions.....	3
1.3	Related documents	3
1.4	Terminology.....	3
1.5	Requirements	3
1.6	Acknowledgements.....	3
1.7	Outline.....	3
2	Installation.....	4
3	Installation Script.....	4
3.1	OsSql modification.....	5
3.2	ABF application configuration.....	5
3.2.1	ABF application setup.....	5
3.2.2	Ingres Date Format.....	5
4	Overview.....	5
5	The IDL2ABF tool.....	6
6	Getting Started	6
6.1	IDL generation	6
6.2	Importing into ABF.....	7
6.3	Compiling a test client/server in ABF.....	7
6.4	Setting up the Broker	8
6.5	Testing the application.....	8

1 Preface


This document is a guide to installation of, and development with, the AGETOR® ABF Application Integration.

1.1 Audience

This guide is written for developers of ABF/Ingres applications who use the AGETOR® ABF Application Integration.

1.2 Typographic conventions

- Text marked with *italics* refer to other publications or definitions of concepts
- Text marked like `AbstractClassName`, `identifier`, `myCoolFunction()` and `cmd` refer to executable commands, identifiers or literal code excerpts

 Issues requiring your special attention are presented like this!

1.3 Related documents

Other documents related to AGETOR® ABF Application Integration are

- *IDL guide*. General description of the IDL language.
- *ARE guide*. Detailed information on setting up and using the AGETOR® Broker

1.4 Terminology

In this document we will use the shorter name *ABF AI* for *AGETOR® ABF Application Integration*.

1.5 Requirements

Before reading this guide, you should be acquainted with the IDL language, and the AGETOR® Broker (see the above-mentioned guides).

1.6 Acknowledgements

The software described in this document includes software developed by the Apache Software Foundation (<http://www.apache.org>).

1.7 Outline

This document focuses on installation of ABF AI and on the creation/generation of ABF services and clients.

2 Installation

Before you begin the installation of the ABF AI make sure that you have the following prerequisites installed:

- Java Development Kit (JDK) version 1.4.2 or newer
- AGETOR Development Kit (ADK) version 2.0.11 or newer
- Ingres version 2.6 or newer
- C language preprocessor.

Download the newest version of ABF AI and copy the package into your “AGETOR_HOME/install/packages” directory. If you are installing a newer version of ABF AI on top of an older or just updating it then the configuration should already be correct. Notice though that new properties or environment variables could have been added and therefore you should always consult the release.txt for changes.


- In windows: Open a command window with `AGETOR_HOME/bin/prompt.bat` and run `installer.bat`
In Unix: Open a shell and setup the environment with `. AGETOR_HOME/bin/proj` and run `installer`.
- After the AGETOR® installtool has started open a Internet browser and point to `http://localhost:8020`
- If prompted for login and password, please type in your login and password (the default is admin/admin)
- Under “Product(s) ready to install” click on “AGETOR ABF Application Integration 2.0.0” and answer the questions
- When the Install Tool has finished, you must run an additional installation script to complete the installation. This is described in detail in section 3.

3 Installation Script

The Ant script that handles local environment configuration after the installation is located in:

```
AGETOR_HOME/install/production-scripts/ABF_AI_2.0.0.ent
```

The actions carried out by the script are described below.

 Text in parentheses () are platform dependent.

1. Locate `AGETOR_HOME` and `II_SYSTEM` environment variables.
2. Rename `II_SYSTEM/ingres/bin/oslsq(.exe)` to `oslsq(.exe)`
3. Copy `AGETOR_HOME/abf_ai/scripts/oslsq(.bat)` to `II_SYSTEM/ingres/bin/oslsq(.bat)`
4. Add `ABF_INCLUDE` to `AGETOR_HOME/bin/agetor(.bat)`
5. Locate platform dependent `libWA` and copy to `II_SYSTEM/ingres/lib`

6. (Unix/linux only) add `II_SYSTEM/ingres/lib` to `LD_LIBRARY_PATH`
7. Add `II_SYSTEM\ingres\lib` to `LIB` envvariable in `agetor(.bat)`
8. Content of `AGETOR_HOME/app/abf/lib/abf.opt` is replaced with a platform specific path.

3.1 Osql modification

The installation script renames the original ingres osql executable to `osqlqr`. It then replaces `osql` with a script calling `osqlqr` after preprocessing the sourcecode to support include and template functionality. These modifications rely on third party programs and if problems should arise from this, the `osql` script can be found in `AGETOR_HOME/app/abf_ai/scripts`

3.2 ABF application configuration

3.2.1 ABF application setup

ABF applications have a predefined source code directory and an optional link-options-file. These two values must be configured for ABF AI to work, the default is:

source code directory for the ABF application set to `AGETOR_HOME/app/abf`

link-options filename points to `AGETOR_HOME/app/abf/lib/abf.opt`

If these values are changed it is important to remember that Ingres specify the source location of include files, namely `.t` and `.h` files created by the generator. These must still be placed in the `AGETOR_HOME/app/abf` as the include directory is set using the environment variable `ABF_INCLUDE` defined in the `agetor` script.

3.2.2 Ingres Date Format

Ingres supports a number of different date formats to fit the standard of individual countries. Communication of dates requires the format to be unique and therefore if the ABF AI requires the date format to be set to german. This can be done with the following command:

```
Ingsetenv II_DATE_FORMAT german
```

The date type in Ingres requires some conversion which is included in the C procedure:

```
AGETOR_HOME/app/abf/w_idato2wa.sc
```

This file must be located in the ABF application source directory.

4 Overview

The main purpose of ABF AI is to ease the development of client/server applications in ABF. This task requires the `idl2abf` tool, which generates stub and skeleton ABF code along with an ascii file for importing structures defined in the IDL into Ingres.

5 The IDL2ABF tool

The `idl2abf` tool is started by typing `idl2abf` in an AGETOR® prompt. More information is available from within the `idl2abf` program itself.

```
C:\Agetor>idl2abf
Usage:

idl2abf <idl-file> [-d<outputpath>] [--old] [--source<path>]

<idl-file>           The name of the file containing the IDL definition
                    without suffix.
-d<outputpath>      Defines optional destination path.
--source<path>     Defines optional source/include path.
--old                Generation of old stub,skeleton and sql file.

Example:
C:\Agetor\app\idl>idl2abf testtypes
```

This will generate the following 6 files in the directory `C:\abfsource\`

```
Modulename_ingresrecords.txt
modulename_marshallng.h
testtypes_skeleton.t
testtypes_stub.h
testtypes_client.osq
testtypes_server.osq
```

Default output directory is `AGETOR_HOME/app/abf`

The `modulename_ingresrecords.txt` is a tabular separated ascii text file with contains the structs defined in the `idl`-file and AGETOR C-procedures for network communication for importing into Ingres.

`modulename_marshallng.h` contains the marshalling procedures for structs and sequences defined in the `idl`-file.

`testtypes_stub.h` contains clientside procedures for communication meant for inclusion in an ABF client side procedure/program (`testtypes_client.osq`, a simple test client, is an example of this).

`testtypes_server.t` is a server template wrapping all procedures defined in the `idl` aswell and initial communication. (`testtypes_server.osq`, a simple test server, is an example of this).

6 Getting Started

6.1 IDL generation

This section describes how to get started using the provided `abf` development tools. To do this we will use the accompanying `testidltypes.idl` file in ABF AI.

- Verify Ingres is running and a database is accessible.

- Start an Agetor prompt/shell.
- From AGETOR_HOME directory change to AGETOR_HOME/app/idl
- Run the idl2abf command with an idl file. (an idl is located here: app/abf_ai/idl/testidltypes.idl)

This generates the 6 files described in section 5.

*.t and *.h files are meant for inclusion and must therefore lie in the predefined include path (ABF_INCLUDE section 3.2.1).

The generated .osq files should be moved into the Ingres specified source code directory if it differs from the current location.

6.2 Importing into ABF

To import the generated records and procedures into an existing database application use the command:

```
iiimport DATABASENAME APPNAME -user=USERNAME -intfile=..._ingresrecords.txt
```

Replace DATABASENAME, APPNAME, USERNAME with appropriate values for ingres.

This will update the database application with the new records and procedures.

6.3 Compiling a test client/server in ABF

Next step is to start developing actual applications. To ease this development a test client and server has been auto generated. To compile these they must be added to the abf application. For these procedures to work the database application defaults must be altered to use the generated source code and C-ORB library.

Check that the source code directory for the ABF application set to AGETOR_HOME/app/abf

Check that the link-options filename points to AGETOR_HOME/app/abf/lib/abf.opt

Create 2 procedures in ABF (client and server) and point their source path to the two .osq files (_client.osq and _server.osq).

Now we should be able to compile the application with the following command:

```
imageapp -uUSERNAME DATABASENAME APPNAME
```

This should (if no errors occurred) generate an executable with the name APPNAME, which can start either the server or the client procedure from the prompt.

For example:

```
APPNAME testtypes_client
```

This will start the test client for the testtypes.idl.

6.4 Setting up the Broker

The only thing left is configuring the right input/output ports for AGETOR broker.

Add the service the broker.cfg located in AGETOR_HOME/conf/broker.cfg:

```
<SERVICES>
  <SERVICE NAME="testtypes" PORT="9999"/>
</SERVICES>
<ENVIRONMENT NAME="agetor">
  <QUESTION NAME="testtypes" QNO="7003"/>
</ENVIRONMENT>
```

The service name and question name must be identical. QNO is is number defined in the idl, and PORT is the port number used for communication. This can be configured from the server command line or by altering the generated source code.

Once the broker.cfg has been updated, it must be started with the broker-start command from an agetor prompt.

6.5 Testing the application

Once the application and been compiled successfully and the Agetor broker configured correctly, it is time to test the application. In the following test we will be using the generated test client and test server along with the AGETOR broker. The test should reveal if the communication between client and server works as expected or if there was problems during data communication between two. We proceed as follows, note that each command requires a separate AGETOR prompt/shell:

Start the AGETOR Broker as described in 6.4 with the command:

```
broker-start
```

Start the test client as described in 6.3 using the command:

```
APPNAME testtypes_client
```

Start the test server with the command:

```
APPNAME testtypes_server
```

Now we should be able to see the server receiving telegram from the client, and once the client is done it will exit, and the test is completed successfully.

🔔 The test client is configured to use localhost, this must be altered manually in the generated source code for a different host.

The server can be killed using the AGETOR tool `bcmd` as follow:
Start the AGETOR `bcmd` with the command:

```
bcmd  
  
show  
  
kill(testtypes)
```

This will cause the server to exit successfully as well and the test is completed.