Interface description

API Shared Library (ewAPIService)

Description of the ewMII

Index of changes

Version/Date	Edited by	Comments / changes
0.1 / 04/22/2006	Meilinger	Original version
0.2 / 04/26/2006	Rohde	Layout of the first draft
1.0 / 05/16/2006	Meilinger	Release version
1.1 / 08/03/2006	Meilinger	Multithreading
1.2 / 12/18/2006	Rohde	Correction “ListNS.sh“
2.0 / 11/20/2007	Rohde	Update for Release 6.0

State: **RELEASED**

Table of Contents

1 ewMII (easy works manufacturing integration interface)

1.1 Preliminary comments:

Since release 5.00, delivery includes a new API shared library. This is named ewMII and completely replaces the previously used shared library named EWApi.

The replacement was made in the following individual files, which were previously created by iTAC: "EWApi.dll" and "EWApiMIC.dll" for Windows systems, "libEWApi.so", „libEWApiMIC.so" and "libEWApiTAO.so" for Linux systems and "EWApiTAO.a" for OS-9 systems.

1.2 Motivation for developing the new API library:

- Use of the new iTAC middleware with all features as they are used in all other Java-based components produced by iTAC.MES.Suite.
- Elimination of native, C++ based, CORBA implementations on the client side. The following individual products are no longer used:
    - VisiBroker C++ for Windows
    - VisiBroker C++ for Linux
    - MICO for Windows
    - MICO for Linux
    - TAO for Linux
    - TAO for OS-9
- Implementation of session handling (apiLogin, apiLogout, apiHeartbeat)
- Considerably simplified memory management for applications which use the new shared library.
- Simplified configuration and initialization of the new shared library using configuration files.
- Logging/debugging, including performance measurement, are supported and can be activated/deactivated at any time.
- Improvement in the code quality and robustness of the new shared library (due in part to the partially automatic generation of C code from the CORBA IDL and the use of the iTAC middleware).
- The new shared library can be used both on VisiBroker servers which do not yet run in the new middleware operating mode (up to and including easy works release 4.21) as well as on servers which are operated with VisiBroker, JavaORB or JacORB in the new middleware!

2 Architecture of the shared library:

On the communication side, the new shared library is based on the iTAC middleware framework. This is developed 100% in Java and is independent of certain CORBA implementations. Through the use of this framework, the shared library can not only fully access the framework's new features, but, in particular, it also profits from the extensive tests which the framework is subjected to on each release. This procedure considerably reduces the error susceptibility of the code and the times which are required for testing the shared library. Having a particularly positive impact with respect to the predecessor version of the shared library is the elimination of the 6 different CORBA implementations.

The use of the iTAC middleware requires a Java environment on the API client (JRE). The connection of the native shared library to the iTAC middleware is realized by means of JNI (Java Native Interface).

Delivery of the new shared library includes a suitable version of the JRE. This JRE is NOT installed on the operating-system level, but is instead located in the file system below the shared library folder. As a result, there are no negative interactions with other Java versions which may already be installed on the computer.

3 System requirements:

The following system requirements must be met in order to use the shared library:

- min. 128 MByte memory
- min. 600 MHz CPU (Pentium II)
- 60 MByte free disk space (library and JRE)
- Windows: NT 4, 2000, XP or 2003
- Linux kernel: 2.2, 2.4 or 2.6

4 Installation and configuration:

The shared library is delivered as a ZIP file. There are separate ZIP files for both Windows and Linux.

- ewMII_Windows_6.00_<builddate>_<buildtime>.zip
- ewMII_Linux_6.00_<builddate>_<buildtime>.tgz

After extracting the archive, a folder named "ewMII/" is created in the current directory. This folder contains the following files and directories:

Windows:	Linux:
`develop/`	`develop/`
`java/`	`java/`
`lib/`	`lib/`
`ewMII.dll`	`libewMII.so`
`ewMII.lcf`	`ewMII.lcf`
`API_Interface_R_6_0_DE.pdf`	`API_Interface_R_6_0_DE.pdf`
`API_Interface_R_6_0_EN.pdf`	`API_Interface_R_5_0_EN.pdf`
`ewMII_DE.pdf`	`ewMII_DE.pdf`
`ewMII_EN.pdf`	`ewMII_EN.pdf`
`ewMII.options`	`ewMII.options`
`ewMIITestClient.exe`	`EWApiTestClient`
`middleware.properties`	`middleware.properties`
`ListNS.cmd`	`ListNS.sh`
`setEnv.cmd`	`setEnv.sh`

The file **"ewMII.dll"** or **"libewMII.so"**, respectively, makes available the API shared library.

The file **"ewMII.options"** is used to configure shared library. In addition to the Java class path to the required iTAC middleware files, the debug logfile can also be specified here and debug logging activated. Moreover, additional Java VM parameters and Java properties can be set here as necessary.

If, for example, the shared library is to be operated on a server which does not yet run in the new middleware (all servers <= release 4.21), the VisiBroker-specific properties are to be set here as well
(-Dvbroker.agent.port=<port>, -Dvbroker.agent.addr=<ipaddress> etc.). Furthermore, the classpath property must be extended in this file with the "vb52_all.jar" archive and the file must be present on the client (it is recommended to copy it from the lib directory on the server).

Note here that the "itac.middleware.orbtype" property in the "middleware.properties" file (see below) is to be set to "EasyORB"! This ensures downward compatibility with the proprietary VisiBroker mechanisms.

Optionally, the name of the APIServer service can also be changed in this file using the "itac.ewmii.service.name" property (e.g. necessary when using the EWApiDemoServer). If this property is not specified, the value "EWApiServices" is used as the default.

The iTAC middleware is configured via the mechanism which is usually used in iTAC.MES.Suite. All relevant settings are made in the **"middleware.properties"** file. The **"ewMII.lcf"** file is used to configure logging of the iTAC middleware framework.

We will not go into detail on the iTAC middleware framework at this point. The three most important settings which need to be made are, however, the following:

```
itac.middleware.orbtype=JavaORB
itac.appid=ewMII
itac.middleware.defaultpool=EwPool
itac.middleware.codebase=http://easyworksserver:8080/
```

1. The "itac.middleware.orbtype" property is used to define the CORBA variant which the middleware uses for the shared library. Possible settings here are "JavaORB", "VisiBroker" and "JacORB". This setting should remain on "JavaORB" (default).

   The type of ORB for the shared library can be selected independent of the ORB type of the server. Thus, the "JavaORB" setting is correct for the shared library even if the server is operated with e.g. the "VisiBroker" ORB type!

2. The "itac.appid" property defines the description of the application. The default setting here is "ewMII". **A more precise description of the client can be specified here (no blanks, underscores or special characters are permitted in this ID).** These descriptions can simplify later evaluations on the server side (e.g. measurement of API runtimes and call frequencies).

   Examples of possible AppIDs:

   ```
   itac.appid=ApiLinie5PC17
   itac.appid=ApiLinie2FKT2
   itac.appid=ApiPQ46ICT3
   ```

3. The "itac.middleware.defaultpool" property is used to define the name of the server pool. The server pool identifies a group of servers which operate in a failover group. Meaningful values include e.g. "ProductivePool" or "TestPool" or "ProductiveSite5“ etc. Which PoolID needs to be set is defined by the server configuration.

4. The "itac.middleware.codebase" property must be used to specify the URL of one of the running servers. On initial start of the shared library, this server must be available. Even though only one server of the server pool is specified here, the client knows all other servers in the pool after the connection has been established for the first time.

The "**setEnv.cmd**" or "**setEnv.sh**" file contains the paths to the required libraries. This applies to both the paths or library paths to the API shared library and the required libraries in the Java directory.

As a test program for the API shared library, the executable program "**ewMIITestClient.exe**“ (Windows) or "**ewMIITestClient**" (Linux) is included in the delivery. Following execution of the "setEnv.cmd" or "setEnv.sh" script in a DOS window or Linux shell, respectively, the test client can be started in the same DOS window or the same Linux shell in this directory.

The source code for the test client is located in the "**develop/**" directory. In addition, this directory contains all files necessary for developing an application which can use the shared library.
The "**lib/**" directory contains the JAR files which contain the iTAC middleware framework and the classes for the interface of the API server. Moreover, additional, ORB type-specific classes may also be located here (e.g. VisiBroker or JacORB JAR files when using these ORB types).

Located in the "**java/**" directory is a complete J2RE version which is compatible with the API shared library.

ListNS.cmd or **ListNS.sh** is a tool which displays the content of a CORBA name server. This tool displays all known services (e.g. for checking whether an API server is accessible).

Example:

```
ListNS.cmd –NScodebase http://easyworksserver:8080/
```

Lists the contents of the name server which is running on server "easyworksserver:8080". Here, either the same URL as is set in the "middleware.properties" file or a different, valid URL of a server in the same server pool should be used.

5 Changes for users of the predecessor version EWApi:

5.1 General:

- The shared library is designed as a "pure" C shared library. All functions comply with the C calling convention (the library can, nevertheless, also be called from C++ programs).
- All function names begin without a leading underscore.
- There exists only one C variant of every function; the previous "_c" postfix has been dropped for this reason.
- The shared library is reentrant and suitable for multithreading.
- Each function which works with arrays as parameters is assigned a "size" parameter for each array (this corresponds approximately to the "numberOfRecords" parameter used previously). This has now been implemented consistently for all API functions.

5.2 Initialization, shutdown and session handling:

- The current version of the shared library can be queried with the "apigetversion" function.
- The shared library is initialized with the "apiinit" function. The only argument which can be specified is an optional path to the configuration file (Options-File-Path). If this is not specified (NULL), the "`ewMII`.options" file is searched for by default in the current working directory. The shared library is not available until "apiinit" has been successfully called.
- "apishutdown" deactivates the shared library. The shared library can be used again only following a subsequent "apiinit" call. **It makes sense to call "apiinit" once only when starting the application and to use the "apishutdown" function only prior to exiting the application. There is no reason to perform multiple initializations and shutdowns during program execution! Moreover, the initial initialization of the shared library takes a relatively long time.** Such a procedure does not have a negative effect on the behavior of the shared library, however.
- Before data-related functions in the shared library can be used, a successful login is necessary in addition to the initialization of the library. To do this, the "apiLogin" function must be used. The positive output value from this function represents the so-called session ID. This ID is required each time any data-related API function is called; it is to be passed as the first parameter to these functions. Multiple simultaneously active logins are possible. Note here that the technical functions are executed with the corresponding session IDs. Before ending the application, all sessions should be ended with an "apiLogout" and then an "apishutdown" executed.

5.3 Error handling

- The function's output value contains the error code (as in the older versions of the shared library). The additional output value arguments used in the predecessor version, appErrorCode and commErrorCode, have been eliminated. The output value of the function represents the sole error code. Depending on the value of the error code, it is possible to differentiate between correct calls, calls with data-related errors and so-called technical errors.

  Range: **`0`** to **`100`**
  The function call was correctly executed on the server side and no error situations occurred.

  Range: **`-1`** to **`-99`** and **`-200`** to **`-399`**
  The function call was executed on the server side, however data-related errors occurred (e.g. invalid station number; serial number not released; work order not activated; etc.)

  Range: **`-100`** to **`-199`**
  The function call was either NOT executed on the server side or technical errors occurred before or after the call was executed (e.g. shared library not initialized; server not accessible; no more memory available etc.).

- Unlike in the predecessor version, the error text is not returned via two separate strings, but rather only one. For this purpose, the last parameter of each function is a pointer to the errorString. This error text is set by the shared library.

5.4 Memory management

- The current version of the shared library generates all memory necessary for "out" parameters itself (and, therefore, for "inout" parameters as well). In the predecessor version, the application itself had to allocate memory for all "out" parameters; this is no longer necessary.

- As a result, all calls have been eliminated which were previously used only to define the size of an "out" array in advance, after which the function was called again with the appropriate amount of reserved memory. This was realized by setting the "numberOfRecords" parameter to "-1" on the first call. **This step has been completely eliminated in the new shared library!** The array size is now specified with the "size" parameter, which is available for each array. A "size" of -1 is no longer permissible.

- After calling an API function and processing the "out" parameters in the application, the memory (requested by the shared library) must again be released. This occurs by calling a single function: "apifreebuffers". It is no longer necessary to separately release each individual memory area as was necessary in the predecessor version of the shared library. The library manages all requested memory itself and can, therefore, again release memory areas which are no longer needed by the application.

  IMPORTANT: Each time the "apifreebuffers" function is called, all buffers requested by the shared library for "out" and "inout" parameters from all previous function calls are deleted WITH THE SAME THREAD! If it is necessary for an application to access "out" parameters from previous functions, these values must either be stored in their own buffers or no "apifreebuffers" calls may be performed during this period!