Data Archiving for Adabas

Application Programming Interface

Version 1.7.1

November 2016

This document applies to Data Archiving for Adabas Version 1.7.1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ADR-REF-171-20161124

Table of Contents

Application Programming Interface

Data Archiving for Adabas provides an application programming interface which allows you to control archive and transfer operations directly from your own infrastructure. The API is supplied for Natural and for the C programming language.

This document consists of the following sections:

- *API for Natural*
- *API for the C Programming Language*

1 API for Natural

Data Archiving for Adabas provides an application programming interface for Natural which enables two interfaces:

Batch Interface
This interface allows you to control, monitor and report archive and transfer activities directly from your own infrastructure.

To-Do List Interface
As an alternative to using Extraction Syntax for data selection, this interface allows you to select data to be archived or transferred using your own custom applications by maintaining business objects (collections of related Adabas ISNs) in a repository file. An Action can then be run which references these business objects and performs the appropriate archiving/transfer operations against the corresponding records. Please refer to *Data Selection Mode* for information on how to define an Action to run in To-Do List mode.

Installing the API

The application programming interface for Natural is included in the SYSADR Natural library provided as an INPL file and supplied as part of the standard installation.

Instructions for installing the INPL file can be found in the readme.txt file located in the "adr/vvrs/examples/natural" directory, where vrs relates to the corresponding Data Archiving for Adabas version.

Implementing the API

The application programming interface for Natural is implemented as a callable sub-program (ADRAPI) with an interface block passed to it as argument.

The interface block (ADRAPI-INTERFACE) is mapped by ADRAPIP1 and constant values used as flags are defined in ADRAPIL1. Both these data areas must be included in your program as LOCAL data.

```
DEFINE DATA
  LOCAL USING ADRAPIL1
  LOCAL USING ADRAPIP1
LOCAL
```

Upon successful completion of an API function ADRAPI-ERROR-CODE will contain 0000. If an API function cannot execute successfully, ADRAPI-ERROR-CODE will contain an error number, whilst ADRAPI-ERROR-DESCRIPTION and ADRAPI-ERROR-DESCRIPTION2 will contain textual descriptions of the error. Refer to the Messages and Codes section for a full list of error codes and their meanings.

Note: Further information about all the fields in the interface block and their particular use in each of the functions is provided in the README text member in library SYSADR.

API Functions

The application programming interface for Natural provides two interfaces that allow you to drive archiving and transfer operations.

- Batch Interface
- To-Do List Interface

Batch Interface

The batch interface allows your own custom application to perform the following functions:

Function	Description
ACTIVITY-INFO	Returns detailed information about a current Activity.
ACTIVITY-LIST	Returns individual lists of Activity IDs, Databases, and Computers for all current Activities.
ARCHIVE	Starts an Activity for a specified Action.
COMPUTER-STATUS	Returns status-related counts for all current Activities processing on one or more specified computers.
DATABASE-STATUS	Returns status-related counts for all current Activities processing on one or more specified databases.
PAUSE	Pauses a current Activity.
RECALL	Recalls a particular Archive for a specified Action.
RESTART	Restarts a failed Activity.
RESUME	Resumes a paused Activity.
STOP	Stops a current or paused Activity.

To-Do List Interface

The To-Do list interface allows your own custom application to perform the following functions:

Function	Description
CLOSE	Closes the To-Do list, allowing the user to leave it in a state where it can be modified (INSERT, ERASE, REPLACE) or submits it ready for processing by the archiving service ready for when it's next commanded to run.
CREATE	Creates a new empty To-Do list for a specific Group, Plan, and Action.
ERASE	Deletes an existing business object (collection of related ISNs) from an open To-Do list.
INSERT	Inserts a new business object (collection of related ISNs) into an open To-Do list.
OPEN	Opens an existing To-Do list for modification (INSERT, ERASE, REPLACE).
REMOVE	Deletes an existing To-Do list regardless of its state.
REPLACE	Replaces an existing business object (collection of related ISNs) in an open To-Do list.
TO_DO Status	Return the status of a To-Do list.

Examples

For a working example of the Batch interface refer to the APICMD program in the SYSADR Natural library.

For a working example of the To-Do List interface refer to the TODOLIST program in the SYSADR Natural library.

Messages and Codes

Refer to the section *API for Natural Error Codes*.

2 API for the C Programming Language

Data Archiving for Adabas provides an application programming interface for the C Programming Language which enables three interfaces:

Batch Interface

This interface allows you to control, monitor, and report archive and transfer Activities directly from your own infrastructure.

To-Do List Interface

As an alternative to using Extraction Syntax for data selection, this interface allows you to select data to be archived or transferred using your own custom applications by maintaining business objects (collections of related Adabas ISNs) in a repository file. An Action can then be run which references these business objects and performs the appropriate archiving/transfer operations against the corresponding records. Please refer to *Data Selection Mode* for information on how to define an Action to run in To-do list mode.

User Library Interface

This interface allows you to create your own record selection plug-in. By implementing the User-Lib interface in your own custom DLL/Lib, it will be auto-discovered by the Data Archiving Service and offered as a data selection mode in the User Interface. When this mode is selected, the custom DLL/Lib will be called to provide the appropriate business objects (collections of related Adabas ISNs) for processing. Please refer to *Data Selection Mode* for information on how to define an Action to run in User Library mode.

Installing the API

The application programming interface for the C programming language is provided in the form of a DLL/Lib and supplied in the product's sdk directory as part of the standard installation.

Implementing the API

For detailed information on developing applications using the application programming interface refer to the adr_api.pdf document located in the product's sdk/doc directory.

API Functions

The application programming interface for the C programming language provides three interfaces that allow you to drive archiving and transfer operations.

- Batch Interface
- To-Do List Interface
- User Library Interface

Batch Interface

The batch interface allows your own custom application to perform the following functions:

Function	Description
ACTIVITY-INFO	Returns detailed information about a current Activity.
ACTIVITY-LIST	Returns individual lists of Activity-IDs, Databases, and Computers for all current Activities.
ARCHIVE	Starts a current Activity for a specified Action.
COMPUTER-STATUS	Returns status-related counts for all current Activities processing on one or more specified computers.
DATABASE-STATUS	Returns status-related counts for all current Activities processing on one or more specified databases.
LIST	Lists archives in the Vault.
PAUSE	Pauses a current Activity.
RECALL	Recalls a particular Archive for a specified Action.
RESTART	Restarts a failed Activity.
RESUME	Resumes a paused Activity.
STOP	Stops a current or paused Activity.

To-Do List Interface

The To-Do list interface allows your own custom application to perform the following functions:

Function	Description
CLOSE	Closes the To-Do list, allowing the user to leave it in a state where it can be modified (INSERT, ERASE, REPLACE) or submits it ready for processing by the archiving service ready for when it's next commanded to run.
CREATE	Creates a new empty To-Do list for a specific Group, Plan, and Action.
ERASE	Deletes an existing business object (collection of related ISNs) from an open To-Do list.
INSERT	Inserts a new business object (collection of related ISNs) into an open To-Do list.

Function	Description
OPEN	Opens an existing To-Do list for modification (INSERT, ERASE, REPLACE).
REMOVE	Deletes an existing To-Do list regardless of its state.
REPLACE	Replaces an existing business object (collection of related ISNs) in an open To-Do list.

User Library Interface

The User-Lib interface allows your own custom DLL/Lib to perform the following functions:

Function	Description
OPEN	Establishes a connection to the custom DLL/Lib and allows for a series of parameters to be passed to it.
FIRST	The custom DLL/Lib returns the first business object (collection of related ISNs) to the Extractor.
NEXT	The custom DLL/Lib returns the next business object (collection of related ISNs) to the Extractor.
NOTIFY	Notification call-back to allow the custom DLL/Lib to implement some form of transaction management and restart/recovery mechanism based on the state supplied in the call back by the Data Archiving Service.
CLOSE	Terminates the connection to the custom DLL/Lib and allows it perform any tidy up operations etc.
STAT	Retrieves the statistics for an archive.

Examples

For a working example of the Batch interface refer to the example source code in the product's sdk/samples/apicmd directory.

For a working example using the To-Do List interface refer to program adrapi in the product's examples/c directory and refer to the readme.txt in the same directory for details on how to run the program.

For an example of the User Library interface refer to the example source code in the product's sdk/samples/extractor directory.

Messages and Codes

Refer to the adr_api.pdf document located in the product's sdk/doc directory.

3 Exporting long Names from Predict

You can export long names from Predict file defintions to be used in dynamic extraction syntax. The ADRPRD export utility requires LFILE 152 to be set to the location of the archiving configuration file when it is used. Please contact your Software AG support center to determine how to acquire the ADRPRD utility.

To use ADRPRD

- Enter the following:

```
LOGON SYSADR
ADRPRD
EXPORT filename [REPLACE=NO|YES]
```

Where *filename* is the name of the Predict file to be exported. REPLACE= is optional, the default value is NO.

Note: The hyphen character "-" is replaced by the underscore character "_" in the archiving configuration.

Example 1

Export a file named EMPLOYEES-FILE from Predict - it will not be replaced in the archiving configuration if it already exists:

```
LOGON SYSADR
ADRPRD
EXPORT EMPLOYEES-FILE
```

Example 2

Export a file named VEHICLES-FILE from Predict - it will be replaced in the archiving configuration if it already exists:

```
LOGON SYSADR
ADRPRD
EXPORT VEHICLES-FILE REPLACE=YES
```

4 Exporting long Names from Natural DDM

You can export long names from DDM defintions to be used in dynamic extraction syntax. The ADRDDM export utility requires LFILE 152 to be set to the location of the archiving configuration file when it is used. Please contact your Software AG support center to determine how to acquire the ADRDDM utility.

To use ADRDDM

- Enter the following:

```
LOGON SYSADR
ADRDDM
EXPORT [LIBRARY=name] ddm [REPLACE=NO|YES]
```

Where*ddm* is the name of the DDM to be exported. REPLACE= is optional, the default value is NO.

Notes:

1. The hyphen character "-" is replaced by the underscore character "_" in the archiving configuration.
2. LIBRARY= is only required for Natural systems on Windows, Unix and Linux.

Example 1

Export a DDM named EMPLOYEES-FILE from mainframe Natural - it will not be replaced in the archiving configuration if it already exists:

```
LOGON SYSADR
ADRDDM
EXPORT EMPLOYEES-FILE
```

Example 2

Export a DDM named VEHICLES-FILE from Natural on open systems - it will be replaced in the archiving configuration if it already exists:

```
LOGON SYSADR
ADRDDM
EXPORT LIBRARY=SYSADR VEHICLES-FILE REPLACE=YES
```