

Administration Milestones

The administration system is used to administer projects consisting of modules generated on the basis of the frame-gallery dialog frames. The administration system is used to integrate functions into the existing dialog management, assign bitmaps and tool bars, define tables which can be addressed from the functions.

The application-shell administration system supports primarily the development of specialized applications. The administration system can also be used, however, when the specialized application is already in production. So, for example, background processes can be monitored, application structures can be defined, access protection can be implemented on specialized functions, and activities within the application can be monitored.

The following topics are covered below:

- Creating Applications
 - Defining Commands
 - Integrating Functions
 - Issuing Authorizations
 - Processing Help Texts
 - Processing Tables
 - Background Processing
 - Loading and Unloading Data
 - Monitoring at Runtime
-

Creating Applications

To integrate an application into an existing system, various actions should be completed before the development work on the new application can begin.

These actions are also necessary when an application is transferred from the development environment to the test environment or from there to a production environment.

Adding Languages

If an application is required in several languages, you should first check whether all necessary languages are defined in the application-shell.

The following languages are initially available:

- English
- German

The application-shell administration interface, including the necessary Natural dialogs, error texts, help texts and commands already exist in English and German.

If you require the administration system in another language, you can add a new language as explained below. The language you add should also be supported by Natural.

To add a language

1. Translate dialogs and error messages into the target language.
2. Translate the name and DIL text for applications, background procedures, commands, functions, function groups, languages (name only), and object types into the target language. To do so, complete the following steps:
 - Choose the corresponding bitmap.

- Choose the object from the list box in the resulting dialog, and choose the Modify Object button.
 - Choose the Translate button from the resulting dialog.
3. In the resulting dialog, choose the language you want to translate to.
 4. In the lower group box, enter the translation for the name and, if necessary, the DIL text.

Defining an Application

With the help of the application-shell administration system, you can administer several applications in various Natural libraries in parallel.

When a new project is begun, the programs that belong to this project are usually kept in a separate Natural library. Since the functions defined in the application-shell that call these modules are independent of the application, the appropriate functions should be accessible to the new application. This means that the functions should be added to the new application.

To define a new application

1. Choose the Application bitmap.
2. Choose the Add a New Object button from the Browse Applications dialog.
3. Define the following in the Maintain Application dialog: Name, DIL Text, Bitmap, Application Type, Library (not required for sub-applications).
4. Define the following in the "Save object as" dialog: Client, Application ID.

Defining Users

Users who require access to an application should be defined in the authorization system.

Note:

If your application does not use access security, you need not define users.

When other applications are already running with the same application-shell core, first check whether the users are already defined in the application-shell. For this, you can use the Browse function. See the Natural Frame Gallery documentation.

To define a new user

1. Choose the User bitmap.
2. Choose the Add a New Object button from the Browse Users dialog.
3. Define the following in the Maintain User dialog: Name, Allowed or disallowed functions.
4. Define the following in the "Save object as" dialog: Client, User ID.
Authorization is checked through the User ID under which the user is added to the system. The user's name is purely for information.

Defining Commands

Application-shell dialog management works on the basis of command buttons, menu commands and tool-bar buttons. All these alternatives serve to start a particular function.

To start a function, that is, an independent function that consists of a combination of several dialogs, you should specify a command and an object type on which the command is used. For example New (command) customer (object).

All commands can be defined in several languages and can be modified even when the system is in production.

Before you define a new command, first review the existing commands to determine whether one of the current commands can be used.

You thus avoid defining unnecessary commands and decrease the amount of maintenance effort.

Defining Commands

Reviewing Existing Local Commands

You can review the commands that are already defined by using the Browse function.

Adding New Commands

To define a new command

1. Choose the Command bitmap.
2. Choose the Add a New Object button from the Browse Commands dialog.
3. Define the following in the Maintain Command dialog: Name, DIL, Bitmap, Main type.
4. Define the following in the "Save object as" dialog: Client, Command ID.

Renaming Commands

You can rename the commands at any time. So, for example, New can be renamed to Add. Renaming should be performed only when absolutely necessary. If you rename a command, take care to ensure that all affected users are informed.

Integrating Functions

A function generated with a frame gallery frame should be integrated for execution in the dialog supplied by the application-shell. After the necessary commands have been defined, this integration can take place.

The integration procedure consists of defining object types and functions and modifying the structure of the application.

Defining Object Types

Reviewing Existing Object Types

You can review the existing object types with the Browse function.

Adding New Object Types

To define a new object type

1. Choose the Object Type bitmap.
2. From the Browse Object Types dialog, choose the Create a New Object button.
3. Define the following in the Maintain Objects dialog: Name, DIL, Bitmap ,Dialog ID (optional - this is the ID for the Browse dialog).
Be sure to use easily recognizable object names.
4. Define the following in the "Save object as" dialog: Client, Object type ID.

Example: *Customer, Invoice, Article*

Defining Functions

Reviewing Existing Functions

You can review the existing functions with the Browse function.

Adding New Functions

To define a new function

1. Choose the Function bitmap.
2. Choose the Add a New Function button from the Browse Functions dialog.
3. Define the following in the Maintain Function dialog: Name, DIL, object-type ID (should be identical to that specified as the corresponding object's object-type ID), dialog ID (identifies the dialog or subprogram for this function), Type (usually dialog, however, the Delete function is always a subprogram), Bitmap, Commands (e.g. New, Display, Modify).
4. Define the following in the "Save object as" dialog: Client, Function ID.

Defining an Application Structure

Many end users prefer to navigate through dialogs rather than by using commands. This is especially true for a new application.

A dialog structure should be constructed so that each function can be reached from the main application through several sub-applications.

You set up the application structure after you have defined the application and the required functions and object types. Once these are defined, you can then modify the application and define the structure in the Assign dialog.

Reviewing Existing Functions

You can review the existing applications with the Browse function.

Modifying Applications

To modify an application

1. Choose the Application bitmap.
2. Choose the Modify an Application button from the Browse Applications dialog.
3. Define the following in the Maintain Application dialog:

Choose the Assign button.

In the Assign dialog, define the following: Type (determines the processes the application can start, e.g. start a browse, start a function),

Available (determines which applications, functions and object types are available with this application).

Issuing Authorizations

The first step in developing access authorization is to determine whether the application will be accessible to all users or access will be protected.

If you decide to enable access protection, then the next stage is to define the users and the access each user will have in the application.

If you decide that access protection is not required, you need not define users.

The final stage is to define access protection for the specific functions. It is through the function groups that the authorized user is defined for the individual functions.

Application Protection

Once an application has been developed, you should decide which functions will be access protected.

Access security is defined in the user exit ZXUSECON.

Access protection for an application has priority over all functions. For example, the functions of an application with access protection are also initially access protected from all users. On the other hand, the functions of an application without access protection are initially accessible to all users.

Defining Function Groups

The function group is the link from a user to a function.

For each function group, access-protected functions are marked as allowed or disallowed. Specific users are then assigned to the function groups. So, for example, the administrator can allow the members of the Contracts function group access to a function, but disallow access to the same function to the members of the Customer Service function group.

To define function groups, choose the Function Group bitmap.

Protecting Functions

For each function, you should decide whether access protection will be applied.

If you do not implement access protection for a function, each user that is allowed access to the application is also permitted to use the function.

If, however, you do implement access protection for a function, only selected users can access this function (which is defined as allowed in at least one function group).

Depending upon whether or not the application is access protected, the following hierarchy for access protection is applied to each user.

Application with Access Protection

- all functions are disallowed
- except for the functions defined as allowed in the function groups
- except again for the functions defined as disallowed in the function groups

Application without Access Protection

- all functions are allowed
- except for the functions defined as disallowed in the function groups
- except again for the functions defined as allowed in the function groups

Complete functions can be access protected. Subdialogs, however, cannot be explicitly access protected.

To integrate a function into the authorization system

1. From the Maintain Function dialog, select the Access Security check box in the Setup group box.
2. Choose the Function bitmap.
3. Add a function group if it does not already exist.
Choose the Function Group bitmap.
4. Integrate the function in the function group.
Choose the Function Group bitmap.
When the application is access protected, each function should be marked as allowed.
When the application is not access protected, each function should be marked as disallowed.
5. Assign the users who are permitted to execute the function to the function group.
Choose the User bitmap.
The next time the user starts the application or chooses Refresh Initialized Data from the Options menu, the defined access authorization will be applied.

Processing Help Texts

The application-shell uses Microsoft's Help Compiler to process the help text. Please refer to the corresponding Microsoft documentation for more information.

Processing Tables

The application-shell table-maintenance function is used to build information objects on which only maintenance functions are to be carried out. Typical examples are country tables and currency tables. The table system avoids generating a file and creating all the necessary maintenance functions.

Deriving Tables

If an information object requires no more than fifty input fields and four descriptors and only the actions listed below are required to process the information object, the information object can be defined in the form of a table.

- Browse
- Delete
- Display
- Modify
- New

Using tables offers the following advantages:

- The effort in database design is reduced.
- The creation of a file is not required.
- The development of the processing functions is not required, since they are automatically generated when a new table is added. This includes the necessary subfunctions for language translation.

Additional processing for this type of information object is not required. The implementation is limited to the definition of a table in the application-shell table maintenance system.

Defining and Using Tables

Once you have defined a table (with the function New Table), you can then begin processing the table data using the dialog functions generated in the system for processing table data. No intermediate steps are required.

When special functions are in development, table data can be accessed using a standard access module. For information on this procedure, see the Natural Frame Gallery documentation.

Maintaining Table Contents

Tables should be maintained by a defined group of people. Your task as the administrator is particularly important when the tables are used for system control. Status tables and parameter tables should be withdrawn from general access to ensure that unauthorized modifications cannot be applied.

Modifications to table values can in some instances have far-reaching consequences which, at the time of modification, may not be apparent. The maintenance of such sensitive tables should therefore have access protection.

Defining Access Protection for Tables

When you define a table, the required maintenance functions are automatically generated.

Functional access protection for functions that maintain tables can be defined analogously to authorization for other functions.

Renaming Tables

You treat a table in the application-shell just like any other object type, however, when you rename a table you use the function Modify Table. Thereby a difference between table definition (administration maintenance) and table contents (user maintenance) is guaranteed.

Renaming a table requires the same care as when you rename a command.

Special Handling of Table Contents

The table maintenance system itself carries out standard validations, such as checking required fields. If you require additional validation, or if, when entering data, particular formats should be observed, a user program that performs these activities should be allocated.

You define the user program when you add the table.

For further information, see the Natural Frame Gallery documentation.

Background Processing

General

Frequently, functions that require high-volume access to the database are implemented as background processes so as not to delay processing.

The application-shell provides facilities for monitoring background processes.

Creating a Background Procedure

The necessary background procedure for the operating system can be written in the application-shell with the command `New Background Procedure`. Input, such as the name of a printer or the name of a necessary workfile, can be added to the procedure as parameters.

Starting a Background Process

Background processes that are to be monitored with the administration system should be created with the program frame `ZXFBA00C`.

The background process is usually started from an online function.

For further information, see the Natural Frame Gallery documentation.

Displaying the Status of a Background Process

You can use the command `Browse Background Processes` to display the current status (process in waiting queue, or process terminated due to error) of the background processes that are sent from an application that uses the current application-shell core.

Error Handling

If a background process is terminated due to an error, detailed information for that error can be displayed with the function `Browse Background Processes` or `Modify Error Log`. Recommendations on error corrections can also be entered here.

Loading and Unloading Data

General

Loading and unloading data may be necessary for various reasons: migration, porting into another environment, archiving data, etc.

Creating Load and Unload Modules

For application-shell object types that already exist, you need not create load and unload modules. However, for specialized object types, these modules should be created. The procedure for creating these modules is added as a background procedure. For further information, see the Natural Frame Gallery documentation.

Allocating Loading and Unloading Modules

To load or unload data in an orderly manner, you should assign the names of the background program and background procedure that are required for execution. You assign the background program and procedure with the function New or Modify Object Type in the Procedure subdialog.

Selecting Data

You use the function Load Object Type or Unload Object Type to select the data to be loaded or unloaded.

Unloading

The unload procedure is used for archiving data, for migrating from one application shell version to the next or for porting data from one environment to another.

The unload job, a background process, is started with the function Unload Object Type. You also define where the data should be unloaded to.

You can observe the activities of the background process with the function Display Load Log.

Loading

The load sequence should be preceded by an unload sequence. Either archived data or data that was unloaded for the purpose of migration can be loaded.

The load sequence is also a background process. You start the background process with the online function Load Object Type and specify where the data is to be loaded.

You can observe the activities of the background process with the functions Browse Load Logs and Display Load Log.

Monitoring at Runtime

During development and when the finished application goes into production, you need particular tools to monitor processing and to access running transactions to correct error situations.

The activities described below are available to all developers during the development of an application.

Processing Errors

The errors that occur in a background process are always logged.

You can use the function Maintain Error Log to display the errors that have occurred and to enter recommendations on how to correct the errors.

Checking Locks

As a rule, all data records are logically locked while they are being processed. You can use the function Browse Locks to display the data records that are currently logically locked.

Maintaining Temporary Data

You can transfer data from one dialog to another using either the global data area or a preliminary file. You can use the functions Browse Preliminary File and Delete Preliminary File to display an overview of the contents of the current file which contains the provisional copy of the data records currently being processed. The entries that have no references to transactions or locks can be removed.