Catching Errors

This page explains the **Catching Errors** in Bridge context. If you were looking for the same information regarding the PAS Designer, refer to Catching Errors in the Designer guide.

If an error occurs, the default behavior is that the Bridge stops the execution of the current service at this point and sends back an error message. If you modify your UML model to catch errors, this behavior can be altered.

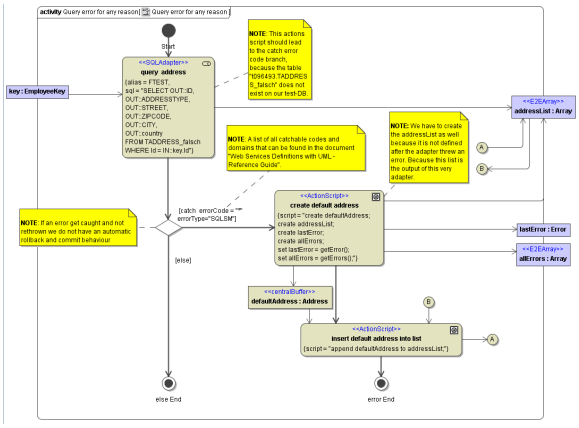
However, fatal errors cannot be caught. They are only written to the log file of the xUML service. Depending on where the fatal error occurs, these errors are not sent back to the client. For example, if a Bridge process starts up and cannot find a valid xUML service, the error is logged and the Bridge process shuts down. Obviously, no client can access this Service. A list of log errors can be found at Log Errors.

The modeler can catch non-fatal errors by using domain and code values. These values are listed on System Errors. They are called **system errors**, because they are returned by the system and not only written to the logging system.

To catch system errors, a junction (decision) object must be drawn. In the guard specification of the transitions starting from the junction symbol, the code and domain as listed in System Errors must be supplied as parameters for the catch statement. Besides using exact code values like **001**, it is also possible to use a wildcard ({}{*}) for all errors of the specified type. Such an example is shown in the following activity diagram.

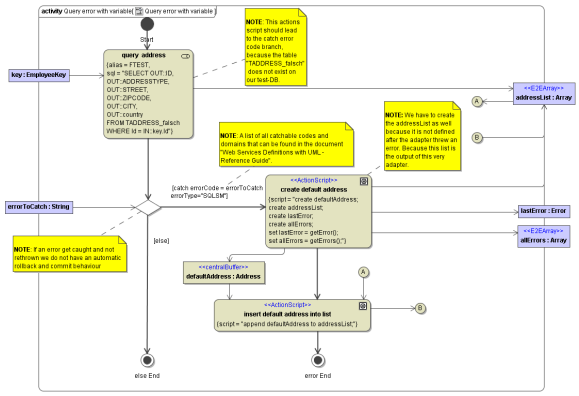
In this example, a default address is sent to the output if the SQL statement fails for any reason (errorCode="*").

Figure: Example of Catching an Error



It is also possible to work with a variable. Therefore, you define the variable and refer to it in the error catch. If the specified error occurs, the Bridge does not abort the request but handles the exception branch. For example:

Figure: Example of Catching an Error Using Variables



On this Page:

- Where Errors Need to be Caught
- Accessing Error Objects

Related Pages:

- getError()
- getErrors()
- Log Errors
- System Errors
- Error Functions
- Modeling Error Handling

Where Errors Need to be Caught

An error can be caught either immediately after the action node that throws the error (for instance an SQL adapter action), or directly after the sub-activity containing the action node.

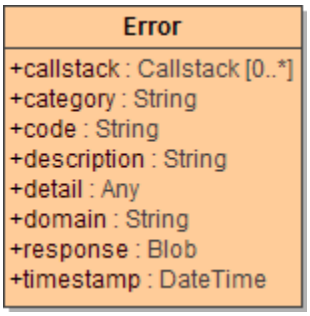
In the latter case, within the activity diagram that implements the sub-activity, any activity step after the error has occurred is not executed in the action flow. As a rule, if the error is not caught in the activity diagram, the Bridge passes the error to the caller (the calling sub-activity). However, in order to catch the error, the junction (decision point) needs to be placed directly after the activity symbol.

Accessing Error Objects

Errors that are caught can be accessed by using the error functions `getError()` and `getErrors()` . See Error Functions for detailed syntax and semantics.

Every error is described using four classifiers as described in Modeling Error Handling. You can find the corresponding attributes in class **Error**.

Figure: Error Class



Additionally, the error object contains information on the **callstack**, some more error **details**, **response** information (e.g. from a SOAP call),and Runtime 2016.4an error **timestamp**.