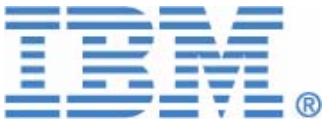
Rational. Statemate

Data Import API

Rational Statemate
Data Import API Reference Manual



Before using the information in this manual, be sure to read the “Notices” section of the Help or the PDF file available from **Help > List of Books**.

This edition applies to IBM® Rational® Statemate® 4.6 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Data Import API Reference Overview

Data Import API Interface

The Statemate Data Import API functions have a C++ language interface. They can be called from C++ language programs, as well as from programs written in other languages that can call C++ functions. This document describes how to import information into the Statemate database from a C++ language program.

Working with the Data Import API

When you use the Data Import API library, you import information into the Statemate database by including calls to various Data Import API functions in your program. An explanation of function calls, parameters, and returned values is provided in the following sections.

After you finish writing and compiling your program, you must link it with the Data Import API Library image. These procedures are explained in **Using Data Import.**

Using Data Import

This section provides information on how to use Data Import functions within a program. It provides information on the following topics:

- **Data Import Function Calls**
- **Function Input Arguments**
- **Using Functions in C/C++ Language Programs**
- **Preparing and Executing Programs**

Data Import Function Calls

Data Import function calls can appear anywhere in your program once an initialization procedure is performed. Here are some examples of valid function calls:

```
stmCreateChart(stm_context, chart_name, chart_type, chart_usage,
short_description, long_description)
```

This function creates a new chart. The chart's page size is 25 x 19. In all other creation functions, when coordinates are given, they should be in this range. For more information, see **stmCreateChart.**

```
stmSetShortLongDescription(stm_context, chart_name, element_name,
element_type, new_short_description, new_long_description)
```

This function sets short and long descriptions for an element. For more inforamtion, see **stmSetShortLongDescription.**

Function Input Arguments

Each argument must be declared to be of a data type recognized by the Data Import library (or by the C compiler). This document includes a complete list of input arguments for each type of database function in the sections that describe the specific function type. refer to the specific section for the lists of arguments relevant for each function.

Using Functions in C/C++ Language Programs

The Data Import has a C++ language interface. To use its functions in a program, you must follow specific procedures to access both the library and your database.

Include Files

Every program that calls Data Import functions must include the definitions for its library data-types and constants. The definitions are contained in the `dataport.h` and `DataimportDefs.h` files.

To incorporate these definitions, include the file by writing the following statement at the beginning of your program:

```
#include dir_name/include/dataport.h
#include dir_name/include/DataimportDefs.h
```

Substitute the value of the environment variable `STM_ROOT` for the `dir_name variable`.

Information Import Process

Perform the following operations when using Data Import functions:

1. First, initialize the import process, via the `stmInitImport` function.
2. Call the Data Import functions to import database information.
3. To finish the information import process, include the following line in your program, after the last Data Import function call:

   ```
   stmCloseImport
   ```

Initializing the Import Process

To initialize the import process, add the following statement to your program before any calls to Data Import library functions:

```
stmInitImport (workarea_dir, project_name)
```

In this call:

- `project_name` - The name of the Rational Statemate project.
- `workarea_dir` - The directory pathname of your workarea in which the specification database is found.

This function returns a `STMContextHandle` pointer if successful, or NULL if unsuccessful.

The following example shows how to initialize the process in a C program. In this example, the user is prompted for the name of the project to open.

```
main( )
{
STMContextHandle *stm_handle;
char     name[32];
char     dir[30];
printf ("Enter name of Statemate project: ");
scanf ("%s", name);
printf ("Enter directory pathname
         for your Workarea: ");
scanf ("%s", dir);
stm_handle = stmInitImport(dir,name);
if (!stm_handle)
  printf ("Init function failed.");
}
```

Note

- The project name (in this case, the content of the variable `name`) is not case sensitive.
- It is recommended that you write the `init` function in the form shown in the example (`stm_handle=...; if(!stm_handle)...;`) to ensure that the `init` function succeeds before continuing.

Preparing and Executing Programs

C++ programs containing Data Import function calls must be linked with the Data Import library. To execute a program containing calls to Data Import functions, follow the procedure for your operating system. The definitions in the `dataport.h` file can be used for debugging purposes.

Windows Systems

Define the environment variable `STM_ROOT`, as follows:

```
SET STM_ROOT=root name
```

Contact your Rational Statemate manager for the name of the root directory of the Rational Statemate tree. For example:

```
SET STM_ROOT=C:\IBM Rational\stmm\4.6
```

Use the following command to compile and link:

```
PROGRAM= my_prog.exe
DLL= <STM_ROOT>\bin\dataimport.dll
DLIB= <STM_ROOT>\lib\dataimport.lib
SRCS= my_prog.c
HDRS= my_prog.h

CFLAGS= /DDLL_LINK /I<STM_ROOT>\include
LIBS= kernel32.lib
all: $(PROGRAM) $(DLL) $(HDRS)

$(PROGRAM): $(SRCS) $(DLIB)
cl $(CFLAGS) $(SRCS) $(DLIB) $(LIBS)

clean:
 -del $(PROGRAM) >nul: 2>&1
 -del *.obj >nul: 2>&1
 -del *.pdb >nul: 2>&1
 -del *.ilk >nul: 2>&1
 -del *.mdp >nul: 2>&1
 -del *.opt >nul: 2>&1
```

In this syntax:

- `my_prog.exe`—The name you want to assign to the executable image
- `my_prog.h`—The header file
- `my_prog.c`—The name of the file containing the C program

Use the following command to execute your program:

```
my_prog
```

UNIX Solaris Systems

Define the environment variable `STM_ROOT`, as follows:

```
% setenv STM_ROOT root_name
```

Contact your Rational Statemate manager for the name of the root directory of the Rational Statemate tree.

Use the following command to compile and link:

```
cc -o <program> <otherflags> <myprog.c>  \
   -L$STM_ROOT/lib -ldata_import \
   $STM_ROOT/lib/x_stubs.o \
   -lm -lsocket -lnsl
```

In this syntax:

- `program`—The name you want to assign to the executable image
- `otherflags`—Can include `-g` or `-O`
- `myprog.c`—The name of the file containing the C program

Use the following command to execute your program:

```
program
```

Optional qualifiers, such as debug, can be added in the compile, link, and execute stages. Refer to your operating system reference manuals for the available options.

The DataImport API library on Solaris is delivered as a shared library (.so): `libdata_import.so`

Unsupported Constructs

The following constructs are not supported in this release of the Data Import API:

- Joint dataflows / arrows
- Library components and component instances
- And States

DataImport Functions

This following sections describe a set of functions that allow you to import information to the Rational Statemate database. The import functions have a C++ language interface.

Using this API as follows, your DataImport program:

- Calls an init function to set the project name and workarea
- Calls a series of creation functions to build the desired model
- Calls a close function to finish the session.

As a result, the given workarea is loaded with the newly-created model. No further operation is required. (for example, no need to manually load charts into the workarea).

Each function returns a status code. Examine this code when the function returns.

Creation Functions

In some creation functions, there are optional arguments that may be omitted when calling the function. These arguments are marked with an asterisk (*) in the Argument field.

This section contains the following Creation functions:

- **stmCreateExternalActivity**
- **stmCreateExternalRouter**
- **stmCreateField**
- **stmCreateGDS**
- **stmCreateInfoFlow**
- **stmCreateLifeline**
- **stmCreateMessage**
- **stmCreateOrderInsignificant**
- **stmCreatePartition**
- **stmCreateRouter**
- **stmCreateState**
- **stmCreateSubroutine**
- **stmCreateTimingConstraint**
- **stmCreateTransition**
- **stmCreateTransitionWithPosNote**
- **stmCreateUserType**

stmAddAttribute

Adds an attribute.

Syntax

```
stmAddAttribute (STMContextHandle, chart_name, element_name, element_type,
attr_name, attr_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
element_type	Input	Enumeration stm_element_type	Element type
attr_name	Input	String	Attribute name
attr_val	Input	String	Attribute value

stmAddAttributeField

Adds an attribute to a record/union field.

Syntax

```
stmAddAttributeField (STMContextHandle, chart_name, element_name, field_name,
element_type, attr_name, attr_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Element type
attr_name	Input	String	Attribute name
attr_val	Input	String	Attribute value

stmAddInfoFlowComponent

Adds an element to an information flow.

Syntax

```
stmAddInfoFlowComponent(stm_context, chart_name,InfoFlow_name,
component_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
InfoFlow name	Input	String	Info Flow name
Component_name	Input	String	Component name

stmAddParameter

Adds a formal parameter to a generic chart. If the given chart is not generic, an error status is returned.

Syntax

```
stmAddParameter (stm_context, chart_name, parameter_name, parameter_type,
parameter_mode)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
parameter_name	Input	String	Parameter name to be added
parameter_type	Input	stm_element_ type	stm_data_item / stm_condition / stm_event
parameter_mode	Input	stm_parameter_ mode	stm_in_parameter / stm_out_parameter / stm_inout_parameter / stm_constant_parameter

stmAddSubroutineActionLanguageCode

Adds subroutine action language code.

Syntax

```
stmAddSubroutineActionLanguageCode (STMContextHandle, chart_name, sbr_name,
set_selected_implmnt, locals_num, stm_sb_local, sbr_code)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “Statemate Action Language” will be set as “Selected Implementation”
locals_num	Input	Integer	Number of local variables for the “Action Language” implementation
sb_locals	Input	stm_sb_local*	Array of stm_sb_local, one for each local variable
sbr_code	Input	String	The Subroutine Action Language code

stmAddSubroutineAdaCode

Add subroutine Ada code.

Syntax

```
stmAddSubroutineAdaCode (STMHandleContext, chart_name, sbr_name,
set_selected_implmnt, sbr_code)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “Ada Code” will be set as “Selected Implementation”
sbr_code	Input	String	The Subroutine Ada code

stmAddSubroutineAnsiCode

Add subroutine ANSI code.

Syntax

```
stmAddSubroutineAnsiCode (STMHandleContext, chart_name, sbr_name,
set_selected_implmnt, sbr_code)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “ANSI C Code” will be set as “Selected Implementation”
sbr_code	Input	String	The Subroutine ANSI C code

stmAddSubroutineExternalToolCode

Add subroutine external tool code.

Syntax

```
stmAddSubroutineExternalToolCode (STMHandleContext, chart_name, sbr_name,
set_selected_implmnt, sbr_code)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “K&R C Code” will be set as “Selected Implementation”
sbr_code	Input	String	The Subroutine K&R C code

stmAddSubroutineKRCCode

Adds subroutine KRC code.

Syntax

```
stmAddSubroutineKRCCode (STMHandleContext, chart_name, sbr_name,
set_selected_implmnt, sbr_code)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “External Tool” will be set as “Selected Implementation”
sbr_code	Input	String	The Subroutine External Tool code

stmAddSubroutineTruthTableCode

Adds subroutine Truth Table code.

Syntax

```
stmAddSubroutineTruthTableCode (STMContextHandle, chart_name, sbr_name,
set_selected_implmnt, locals_num, sbr_locals, stm_truth_table_rec)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
set_selected_implmnt	Input	stm_boolean	If stm_true, “Truth Table” will be set as “Selected Implementation”
locals_num	Input	integer	Number of local variables for the “Truth Table” implementation
sbr_locals	Input	stm_sb_local*	Array of stm_sb_local, one for each local variable
tt_rec	Input	stm_truth_table_rec*	Pointer to stm_truth_table_rec with the truth-table data

stmBindParameter

Binds a formal parameter of a generic chart to an actual one.

Syntax

```
stmBindParameter (stm_context, chart_name, instance_box_name,
formal_parameter_name, actual_parameter_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name where the generic instance box exists
instance_box_name	Input	String	The box name of the generic instance
formal_parameter_name	Input	String	Name of formal parameter
actual_parameter_name	Input	String	Name of actual parameter

stmBindParameterWithParamType

Bind parameter of an instance to an actual parameter with type.

Syntax

```
stmBindParameterWithParamType(stm_context, chart_name, instance_box_name,
formal_parameter_name, formal_parameter_type, actual_parameter_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
instance_box_name	Input	String	Instance Box Name
formal_parameter_name	Input	String	Formal parameter name
formal_parameter_type	Input	String	Type of Formal parameter
actual_parameter_name	Input	String	Actual parameter name

stmCreateAction

Creates a named action.

Syntax

```
stmCreateAction (stm_context, chart_name, action_name, definition,
short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
action_name	Input	String	Action name
definition	Input	String	(*)Optional
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional. NULL for no description

stmCreateActivity

Creates a named activity in a specified activity chart with specified coordinates. The new activity will be created with an empty mini spec, selected implementation set to “Best Match” and termination type set to “Reactive Control.” To define mini spec or set a different selected implementation or termination type, a different API call is supplied later in this document.

Syntax

```
stmCreateActivity (stm_context, chart_name, activity_name,
parent_activity_name, min_x, min_y, max_x, max_y, activity_type,
short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activity chart name
activity_name	Input	String	Activity name
parent_activity_name	Input	String	Parent activity name (NULL for no parent)
min_x	Input	Double	Bounding box min x coordinate
min_y	Input	Double	Bounding box min y coordinate
max_x	Input	Double	Bounding box max x coordinate
max_y	Input	Double	Bounding box max y coordinate
activity_type	Input	Enumeration stm_activity_type	Activity type: stm_ac_internal stm_ac_control stm_ac_data_store
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateChart

Creates a new chart. The chart's page size is 25 x 19. In all other creation functions, when coordinates are given, they should be in this range.

Syntax

```
stmCreateChart(stm_context, chart_name, chart_type, chart_usage,
short_description, long_description)
```

Return Type

status: success / fail

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
chart_type	Input	Enumeration stm_chart_type	Chart type: stm_ch_activity stm_ch_state stm_ch_dictionary -GDS
chart_usage	Input	Enumeration stm_chart_usage	Chart usage: stm_ch_usage_normal stm_ch_usage_generic
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateCondition

Create a condition.

Syntax

```
stmCreateCondition (stm_context, chart_name, condition_name, structure,
usage, array_lindex, array_rindex, definition, short_description,
long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
condition _name	Input	String	Condition name
structure	Input	Enumeration stm_co_structure_type	stm_co_single stm_co_array stm_co_missing
usage	Input	Enumeration stmstm_co_definition_type	stm_co_primitive (variable) stm_codi_constant stm_codi_compound
array_lindex	Input	String	(*) Optional Left index of array. May be either a numeric constant or a dataitem name. NULL for non array type
array_rindex	Input	String	(*) Optional Right index of array. May be either a numeric constant or a dataitem name. NULL for non array type
definition	Input	String	(*) Optional Definition of non variable usage. NULL for no definition.

short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

Note

If you are using an enumerated value for your array indices, you should define the enumerated index in full-quote notation form. For example, defining `COLORS'RED` as the left index of an array sets the value `RED` in the **of enum** field of the element's Data Dictionary form.

stmCreateConnector

Creates a connector in a chart.

Syntax

```
stmCreateConnector(stm_context, chart_name, cn_name, cn_type,x_pos, y_pos);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned byStmInitImport
chart_name	Input	String	Chart name
cn_type	Input	Enumeration_stm_ connector_type	Connector type
x_pos	Input	Double	Connector X coordinate
y_pos	Input	Double	Connector Y coordinate

stmCreateDataflow

Create a labeled dataflow between two activities. Source and target coordinates should be on the source and target boxes respectively.

Syntax

```
stmCreateDataflow (stm_context, chart_name, source_activity_name,
target_activity_name, label, source_x, source_y, target_x, target_y)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
source _activity_name	Input	String	Source activity name
target _activity_name	Input	String	Target activity name
label	Input	String	Dataflow label NULL if there is no label
source_x	Input	Double	X coordinate of dataflow source point
source _y	Input	Double	Y coordinate of dataflow source point
target_x	Input	Double	X coordinate of dataflow target point
target _y	Input	Double	Y coordinate of dataflow target point

stmCreateDataFlowWithPos

This function is similar to `stmCreateDataflow`, but allows the user to set coordinates for the data-flow label.

Syntax

```
stmCreateDataFlowWithPos (STMContextHandle, chart_name, source_activity_name,
target_activity_name, label, source_x, source_y, target_x, target_y, label_x,
label_y)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
source_activity_name	Input	String	Source activity name
target_activity_name	Input	String	Target activity name
label	Input	Integer	Transition label NULL if there is no label
source_x	Input	Double	X coordinate of transition source point
source_y	Input	Double	Y coordinate of transition source point
target_x	Input	Double	X coordinate of transition target point
target_y	Input	Double	Y coordinate of transition target point
label_x	Input	Double	(*)Optional
label_y	Input	Dboule	(*)Optional

stmCreateDataItem

Creates a data item.

Syntax

```
stmCreateDataItem (stm_context, chart_name, dataitem_name, structure,
data_type, usage, array_lindex, array_rindex, definition, user_type_name,
short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
structure	Input	Enumeration stm_di_structure_type	stm_di_single, stm_di_array, stm_di_queue
data_type	Input	Enumeration stm_di_data_type	stm_di_integer stm_di_real stm_di_string stm_di_bit stm_di_bit_array stm_di_user_type stm_di_missing
usage	Input	Enumeration stm_di_definition_type	stm_stm_di_primitive (variable) stm_di_constant stm_di_compound stm_di_alias
array_lindex	Input	String	(*) Optional Left index of array. May be either a numeric constant or a dataitem name. NULL for non array type.

array_rindex	Input	String	(*) Optional Right index of array. May be either a numeric constant or a dataitem name. NULL for non array type.
Definition	Input	String	(*) Optional Definition of non-variable usage. NULL for no definition.
user_type_name	Input	String	(*) Optional Name of user defined type. NULL for no user defined type.
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

Note

If you are using an enumerated value for your array indices, you should define the enumerated index in full-quote notation form. For example, defining `COLOR' RED` as the left index of an array sets the value `RED` in the **of enum** field of the element's Data Dictionary form.

stmCreateDefaultTransition

Creates a labeled default transition between the given source point and the given target state. The target coordinates should be on the target box.

Syntax

```
stmCreateDefaultTransition (stm_context, chart_name, target_state_name,
label, source_x, source_y, target_x, target_y)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
target_state_name	Input	String	Target state name
label	Input	String	Transition label NULL if there is no label
source_x	Input	Double	X coordinate of transition source point
source _y	Input	Double	Y coordinate of transition source point
target_x	Input	Double	X coordinate of transition target point
target _y	Input	Double	Y coordinate of transition target point

stmCreateDefaultTransitionWithPosNote

Creates a Default Transition with a label and a transition note between two states. Source and target coordinates should be on the source and target boxes respectively.

Syntax

```
stmCreateDefaultTransitionWithPosNote (stm_context, chart_name,
target_state_name, label, source_x, source_y, target_x, target_y, tr_note,
label_x, label_y, note_x, note_y)
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
target_state_name	Input	String	Target state name
label	Input	String	Transition label
source_x	Input	Double	X coordinate of transition source point
source _y	Input	Double	Y coordinate of transition source point
target_x	Input	Double	X coordinate of transition target point
target _y	Input	Double	Y coordinate of transition target point
tr_note	Input	String	(*) optional transition note
label_x	Input	Double	(*)optional X coordinate of label
label_y	Input	Double	(*)optional Y coordinate of label
note_x	Input	Double	(*)optional X coordinate of transition note
note_y	Input	Double	(*)optional Y coordinate of transition note

stmCreateEvent

Creates an event.

Syntax

```
stmCreateEvent (stm_context, chart_name, event_name, structure, usage,
array_lindex, array_rindex, definition, short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
event_name	Input	String	Event name
structure	Input	Enumeration stm_ev_structure_type	stm_ev_single stm_ev_array stm_ev_missing
usage	Input	Enumeration stm_ev_definition_type	stm_ev_primitive (variable) sDtm_ev_compound
array_lindex	Input	String	(*) Optional Left index of array. May be either a numeric constant or a dataitem name. NULL for non array type.
array_rindex	Input	String	(*) Optional Right index of array. May be either a numeric constant or a dataitem name. NULL for non array type.
definition	Input	String	(*) Optional Definition of non variable usage. NULL for no definition.
short_description	Input	String	(*) Optional NULL for no description

long_description	Input	String	(*) Optional NULL for no description

Note

If you are using an enumerated value for your array indices, you should define the enumerated index in full-quote notation form. For example, defining `COLOR' RED` as the left index of an array sets the value `RED` in the **of enum** field of the element's Data Dictionary form.

stmCreateExternalActivity

Creates a named external activity in a specified activitychart with specified coordinates.

Syntax

```
stmCreateExternalActivity (stm_context, chart_name, activity_name, user_key,
min_x, min_y, max_x, max_y, activity_type, short_description,
long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
activity_name	Input	String	Activity name
user_key	Input	String	A user key should be a unique string to identify occurrences of the same external activity
min_x	Input	Double	Bounding box min x coordinate
min_y	Input	Double	Bounding box min y coordinate
max_x	Input	Double	Bounding box max x coordinate
max_y	Input	Double	Bounding box max y coordinate

activity_type	Input	Enumeration stm_activity_type	Activity type: stm_ac_external stm_ac_environment
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateExternalRouter

Creates a named external router.

Syntax

```
stmCreateExternalRouter(STMContextHandle, chart_name, router_name, min_x,
min_y, max_x, max_y, short_description DEFAULT_VAL(=NULL), long_description
DEFAULT_VAL(=NULL)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
router_name	Input	String	Router name
min_x	Input	Double	Bounding box min x coordinate
min_y	Input	Double	Bounding box min y coordinate
max_x	Input	Double	Bounding box max x coordinate
max_y	Input	Double	Bounding box max y coordinate
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateField

Adds Field To Record.

Syntax

```
stmCreateField(stm_context, chart_name, user_type_name, field_name,
structure, data_type, array_lindex, array_rindex, user_type_name,
short_description, long_description);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Field_name	Input	String	New field name
User_type_name	Input	String	User type (record) name to add field to
Structure	Input	Enumeration stm_dt_structure_type	Data Structure of new field
Data_type	Input	Enumeration stm_dt_data_type	Data Type of new field
Array_lindex	Input	String	(*) optional Left Index of array
Array_rindex	Input	String	(*) optional Right Index of array
User_type_name	Input	String	(*) optional User Type name - in case field's data type was set to user_defined
Short_description	Input	String	(*) optional Short Description for field
Long_description	Input	String	(*) optional Long Description for field

stmCreateGDS

Creates a new GDS (Global Definition Set).

Syntax

```
stmCreateGDS(stm_context, chart_name)
```

Return Type

status: success / fail

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name

stmCreateInfoFlow

Creates an info flow.

Syntax

```
stmCreateInfoFlow(stm_context, chart_name, Info_name, short_description,
long_description);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Info name	Input	String	Info Flow name
Short_description	Input	String	(*)optional short description for info flow
Long_description	Input	String	(*)optional long description for info flow

stmCreateLifeline

Creates a new lifeline in a Sequence diagram.

Syntax

```
stmCreateLifeline(stm_context, chart_name, lifeline_name, x, ll_type);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Lifeline_name	Input	String	Linfeline name
X	Input	Double	Lifeline position
ll_type	Input	Enumeration_stm _element_type	Lifeline type

stmCreateMessage

Creates a message between two lifelines in a sequence diagram.

Syntax

```
stmCreateMessage(Stm_Context, chart_name, source_lifeline_name,
target_lifeline_name, label, source_x, target_x, y, msg_note);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
source_lifeline_ name	Input	String	Source Lifeline Name
target_ lifeline _name	Input	String	Target Lifeline Name
Label	Input	String	Message label
source_x	Input	Double	X coordinate of Meaagse source point
target_x	Input	Double	X coordinate of Meaagse target point
Y	Input	Double	Y coordinate of Message
msg_note	Input	String	Transition note

stmCreateOrderInsignificant

Creates an Order Insignificant Line.

Syntax

```
stmCreateOrderInsignificant(Stm_Context, chart_name, source_y, target_y,
x,label);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
source_y	Input	Double	Y coordinate for Order Insignificant Line source point
target_y	Input	Double	Y coordinate for Order Insignificant Line target point
x	Input	Double	X coordinate of Order Insignificant Line
label	Input	String	Label string for Order Insignificant Line

stmCreatePartition

Creates a Partition line in a sequence diagram

Syntax

```
stmCreatePartition(Stm_Context, chart_name, partition_name, pl_y, name_x);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
partition_name	Input	String	PartitionLine name
pl_y	Input	Double	Y coordinate of Partition Line
name_x	Input	Double	X coordinate of Partition Line name

stmCreateRouter

Creates a new router.

Syntax

```
stmCreateRouter(stmContextHandle, chart_name, router_name,
parent_activity_name, min_x, min_y, max_x, max_y, short_descritption
DEFAULT_VAL(=NULL), long_description DEFAULT_VAL(=NULL);
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
router_name	Input	String	Router name
parent_activity_name	Input	String	Parent state name (NULL for no parent)
min_x	Input	Double	Bounding box min x coordinate
min_y	Input	Double	Bounding box min y coordinate
max_x	Input	Double	Bounding box max x coordinate
max_y	Input	Double	Bounding box max y coordinate
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateState

Creates a named state in a specified statechart with specified coordinates.

Syntax

```
stmCreateState (stm_context, chart_name, state_name, parent_state_name,
min_x, min_y, max_x, max_y, short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
state_name	Input	String	State name
parent_state_name	Input	String	Parent state name (NULL for no parent)
min_x	Input	Double	Bounding box min x coordinate
min_y	Input	Double	Bounding box min xycoordinate
max_x	Input	Double	Bounding box max x coordinate
max_y	Input	Double	Bounding box max y coordinate
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateSubroutine

Creates a subroutine.

Syntax

```
stmCreateSubroutine (STMContextHandle, chart_name, sbr_name, sbr_type,
sbr_ret_type, ret_usr_type_name, sbr_impl_type, params_num, stm_sb_params,
stm_sb_global, short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
sbr_type	Input	stm_sb_definition_type	stm_sb_function/ stm_sd_procedure/ stm_sb_task
sbr_ret_type	Input	stm_sb_return_type	stm_sb_missing/ stm_sb_integer/ stm_sb_real/ stm_sb_string/ stm_sb_bit/ stm_sb_bit_array/ stm_sb_condition/ stm_sb_user_type
ret_user_type_name	Input	String	If return type is user_type, name of this user_type
sbr_impl_type	Input	stm_sb_select_implementation	stm_sb_kr_c_code/ stm_sb_truth_table/ stm_sb_ada_code/ stm_sb_action_lang/ and so forth.
params_num	Input	Integer	Array of stm_sb_param, one for every parameter
sbr_params	Input	stm_sb_param*	Array of stm_sb_param, one for every parameter

Argument Name	Input/ Output	Argument Type	Argument Description
globals_num	Input	Integer	Number of Subroutine global variables
sb_global	Input	stm_sb_global*	Array of stm_sb_global, one for every global variable
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

stmCreateTimingConstraint

Creates timing constraint in a sequence diagram.

Syntax

```
stmCreateTimingConstraint(Stm_Context, chart_name, source_y, target_y,
x,label);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
source_y	Input	Double	Y coordinate for Timing Constraint source point
target_y	Input	Double	Y coordinate for Timing Constraint target point
x	Input	Double	X coordinate of Timing Constraint
label	Input	String	Label string for Timing Constraint

stmCreateTransition

Creates a labeled transition between two states. Source and target coordinates should be on the source and target boxes respectively.

Syntax

```
stmCreateTransition (stm_context, chart_name, source_state_name,
target_state_name, label, source_x, source_y, target_x, target_y)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
source_state_name	Input	String	Source state name
target_state_name	Input	String	Target state name
label	Input	String	Transition label NULL if there is no label
source_x	Input	Double	X coordinate of transition source point
source _y	Input	Double	Y coordinate of transition source point
target_x	Input	Double	X coordinate of transition target point
target _y	Input	Double	Y coordinate of transition target point

stmCreateTransitionWithPosNote

Creates a transition with a label and a transition note between two states. Source and target coordinates should be on the source and target boxes respectively.

Syntax

```
stmCreateTransitionWithPosNote (stm_context, chart_name, source_state_name,
target_state_name, label, source_x, source_y, target_x, target_y, tr_note,
label_x, label_y, note_x, note_y);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
source_state_name	Input	String	Source state name
target_state_name	Input	String	Target state name
label	Input	String	Transition label
source_x	Input	Double	X coordinate of transition source point
source _y	Input	Double	Y coordinate of transition source point
target_x	Input	Double	X coordinate of transition target point
target _y	Input	Double	Y coordinate of transition target point
tr_note	Input	String	(*) optional transition note
label_x	Input	Double	(*) optional X coordinate of label
label_y	Input	Double	(*) optional Y coordinate of label
note_x	Input	Double	(*) optional X coordinate of transition note
label_y	Input	Double	(*) optional Y coordinate of transition note

stmCreateUserType

Create a user defined type.

Syntax

```
stmCreateUserType (stm_context, chart_name, UserType_name, structure,
data_type, array_lindex, array_rindex, definition, user_type_name,
short_description, long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
usertype_name	Input	String	User type name
structure	Input	Enumeration stm_dt_structure_type	stm_dt_single, stm_dt_array stm_dt_queue
data_type	Input	Enumeration stm_dt_data_type	stm_dti_integer stm_dti_real stm_dti_string stm_dti_bit stm_dti_bit_array stm_dti_user_type stm_dtt_enum_type stm_dtt_record_type stm_dti_missing
array_lindex	Input	String	(*) Optional Left index of array. May be either a numeric constant or a dataitem name. NULL for non array type.
array_rindex	Input	String	(*) Optional Right index of array. May be either a numeric constant or a dataitem name. NULL for non array type.

definition	Input	String	(*) Optional Definition of non-variable usage NULL for no definition
user_type_name	Input	String	(*) Optional Name of user defined type. NULL for no user defined type
short_description	Input	String	(*) Optional NULL for no description
long_description	Input	String	(*) Optional NULL for no description

Note

If you are using an enumerated value for your array indices, you should define the enumerated index in full-quote notation form. For example, defining `COLOR' RED` as the left index of an array sets the value `RED` in the **of enum** field of the element's Data Dictionary form.

Set/Modify Functions

This sections contains the following Set/Modify functions:

- stmSetBitLimitsSubLocalVar
- stmSetConditionLimits
- stmSetConditionLimitsField
- stmSetConditionLimitsSubLocalVar
- stmSetEnumTypeLimits
- stmSetHandleErrorFunc
- stmSetIntegerLimits
- stmSetIntegerLimitsField
- stmSetIntegerLimitsSubLocalVar
- stmSetIntegerLimitsSubParameter
- stmSetRealLimits
- stmSetRealLimitsField
- stmSetRealLimitsSubLocalVar
- stmSetRealLimitsSubParameter
- stmSetRouterMode
- stmSetShortDescriptionSubParameters
- stmSetShortDescriptionSubLocalVar
- stmSetShortLongDescription
- stmSetShortLongDescriptionField
- stmSetStringLength
- stmSetStringLengthField
- stmSetStringLengthSubLocalVar
- stmSetStringLengthSubParameter
- stmSetStringLimits
- stmSetStringLimitsField
- stmSetStringLimitsSubLocalVar
- stmSetSTStaticReaction
- stmUserTypeChangeScope

stmActionChangeScope

Changes the definition scope of a given action.

Syntax

```
stmActionChangeScope (stm_context, chart_name, action_name,
new_chart _name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
action_name	Input	String	Action name
new_chart _name	Input	String	New chart name

stmChangeActivityGraphics

Changes the activity's graphics. Set new parent and new coordinates. The new parent can be Null which means the activity is now a top level activity. When no **new_parent_activity_name** is specified, it means no change in parent box. All four coordinates must be given.

Syntax

```
stmChangeActivityGraphics (stm_context, chart_name, unique_activity_name,
new_ min_x, new_min_y, new_max_x, new_max_y, new_parent_activity_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
unique_activity_name	Input	String	Unique Activity name
new_min_x	Input	Double	New Bounding box min x coordinate
new_min_y	Input	Double	New Bounding box min y coordinate
new_max_x	Input	Double	New Bounding box max x coordinate
new_max_y	Input	Double	New Bounding box max y coordinate
new_parent_activity_name	Input	String	(*) Optional New parent activity name

stmChangeChartUsage

Changes usage of a chart.

Syntax

```
stmChangeChartUsage(stm_context, chart_name, new_usage, modify_references);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
New_usage	Input	Enumeration stm_chart_usage	New usage for chart
modify_references	Input	Boolean	(*) optionalChange all references to this chart

stmChangeExternalActivityGraphics

Changes the external activity's graphics. Set new coordinates.

Syntax

```
stmChangeExternalActivityGraphics (stm_context, chart_name,
ext_activity_name, user_key, new_ min_x, new_min_y, new_max_x, new_max_y)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
ext_activity_name	Input	String	External Activity name
user_key	Input	String	User key to identify an external activity occurrence
new_min_x	Input	Double	New Bounding box min x coordinate
new_min_y	Input	Double	New Bounding box min y coordinate
new_max_x	Input	Double	New Bounding box max x coordinate
new_max_y	Input	Double	New Bounding box max y coordinate

stmChangeStateGraphics

Changes the state's graphics. Sets new parent and new coordinates. The new parent can be Null which means no change in parent box. All four coordinates must be given.

Syntax

```
stmChangeStateGraphics (stm_context, chart_name, unique_state_name, name,
new_ min_x, new_min_y, new_max_x, new_max_y, new_parent_state_name)
```

Retrun Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
unique_state_name	Input	String	Unique state name
new_min_x	Input	Double	New Bounding box min x coordinate
new_min_y	Input	Double	New Bounding box min y coordinate
new_max_x	Input	Double	New Bounding box max x coordinate
new_max_y	Input	Double	New Bounding box max y coordinate
new_parent_state_name	Input	String	(*) Optional New parent state name

stmConditionChangeScope

Changes the definition scope of a given condition.

Syntax

```
stmConditionChangeScope (stm_context, chart_name, condition_name, new_chart
_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
condition _name	Input	String	Condition name
new_chart _name	Input	String	New chart name

stmDataItemChangeScope

Changes the definition scope of a given data item.

Syntax

```
stmDataItemChangeScope (stm_context, chart_name, dataitem_name,
new_chart_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
new_chart _name	Input	String	New chart name

stmEventChangeScope

Changes the definition scope of a given event.

Syntax

```
stmEventChangeScope (stm_context, chart_name, event_name,
new_chart _name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Argument

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
event_name	Input	String	Event name
new_chart _name	Input	String	New chart name

stmInfoFlowChangeScope

Changes the definition scope of a given information-flow.

Syntax

```
stmInfoFlowChangeScope (stm_context, chart_name, if_item_name,
new_chart_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
if_item_name	Input	String	Information-flow name
new_chart _name	Input	String	New chart name

stmModifyAttribute

Modifies an attribute.

Syntax

```
stmModifyAttribute (STMContextHandle, chart_name, element_name, element_type,
attr_name, attr_value)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
element_type	Input	Enumeration stm_element_type	stm_chart/ stm_state/ stm_event/ stm_data_item/ and so forth
attr_name	Input	String	Attribute name
attr_value	Input	String	Attribute value

stmModifyAttributeField

Modify an attribute of a record/union field.

Syntax

```
stmModifyAttributeField (STMContextHandle, chart_name, element_name,
field_name, element_type, attr_name, attr_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Field element type
attr_name	Input	String	Attribute name
attr_value	Input	String	Attribute value

stmModifyBasicArrowLabel

Modifies a label of a basic arrow.

Syntax

```
stmModifyBasicArrowLabel (handle, arrow_id, new_label)
```

Return Type

```
int
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	STMContextHandle *	STM context descriptor that was returned by stmInitImport
arrow_id	Input	stm_id	The ID of the basic arrow.
new_label	Input	char *	The new label to be attached to the basic arrow.

stmRenameActivity

Rename an existing activity. The activity name should be a unique name in the chart. For example: A1.A2 if there is more then one A2 in the chart.

Syntax

```
stmRenameActivity (stm_context, chart_name, unique_activity_name,
new_activity_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
unique_activity_name	Input	String	Unique activity name
new_activity_name	Input	String	New activity name

stmRenameChartOrGDS

Renames a chart or a GDS.

Syntax

```
stmRenameChartOrGDS(stm_context, chart_name, new_name, modify_references);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
New_name	Input	String	New Name for Chart or GDS
modify_references	Input	Boolean	(*) optional Change all references to this chart to use the new name

stmRenameExternalActivity

Renames an existing external activity.

Syntax

```
stmRenameExternalActivity (stm_context, chart_name, ext_activity_name,
user_key, new_ext_activity_name, new_user_key)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
ext_activity_name	Input	String	External activity name
user_key	Input	String	User key to identify an external activity occurrence
new_ext_activity_name	Input	String	New externalactivity name
new_user_key	Input	String	(*) Optional New user key

stmRenameState

Renames an existing state. The state name should be a unique name in the chart. For example: S1.S2 if there is more then one S2 in the chart.

Syntax

```
stmRenameState (stm_context, chart_name, unique_state_name, new_state_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
unique_state_name	Input	String	Unique state name
new_state_name	Input	String	New state name

stmSetACMiniSpec

Sets mini spec for an existing activity in a specified activitychart. If a mini spec exists for this activity, it is replaced.

Syntax

```
stmSetACMiniSpec (stm_context, chart_name, activity_name, mini_spec)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
activity_name	Input	String	Activity name
mini_spec	Input	String	Mini spec for the activity

stmSetACSelectedImplementation

Sets selected implementation for an existing activity in a specified activity.

Syntax

```
stmSetACSelectedImplementation (stm_context, chart_name, activity_name,
selected_implementation)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
activity_name	Input	String	Activity name
selected_implementation	Input	Enumeration stm_ac_selected_ implementationString	stm_ac_mini_spec_imp/ stm_ac_subroutine_bind_imp/ stm_ac_truth_table_imp/ stm_ac_none Mini spec/subroutine binding/truth table/best match/none

stmSetACTermination

Sets termination type for an existing activity in a specified activitychart.

Syntax

```
stmSetACTermination (stm_context, chart_name, activity_name,
termination_type)
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Activitychart name
activity_name	Input	String	Activity name
termination_type	Input	Enumeration stm_activity _termination	stm_ac_controlled_termination (Reactive control) / stm_ac_self_termination (reactive self) / stm_ac_procedure_like

stmSetBitArrayIndices

Sets BitArray indices.

Syntax

```
stmSetBitArrayIndices(stm_context, chart_name, data_item_name,
bitarray_lindex,bitarray_rindex);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Data_Item_name	Input	String	BitArray name
bitarray_lindex	Input	String	Left Index of BitArray
bitarray_rindex	Input	String	Right Index of BitArray

stmSetBitArrayLimits

Defines additional parameters for a “Bitarray” data item: left and right indices and default value. An error is issued if the data item is not of “Bitarray” type.

Syntax

```
stmSetBitArrayLimits (handle, chart_name, dataitem_name, element_type,
bitarray_lindex, bitarray_rindex, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
bitarray_lindex	Input	String	Bit array's left bound
bitarray_rindex	Input	String	Bit array's right bound
default_val	Input	String	(*)Optional

Note

This API supports all bit-array elements that may have a default value (except for field, subroutine parameter, and subroutine local variable):

- data_item
- data_type

stmSetBitArrayLimitsField

Defines additional parameters for a “Bitarray” field: left and right indices and default value. An error is issued if the data item is not of “Bitarray” type.

Syntax

```
stmSetBitArrayLimitsField (handle, chart_name, dataitem_name, field_name,
element_type, bitarray_lindex, bitarray_rindex, default_val)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
bitarray_lindex	Input	String	Bit array's left bound
bitarray_rindex	Input	String	Bit array's right bound
default_val	Input	String	(*)Optional

Note

This API supports all element types that may have a “Bitarray” Field:

- data_item
- data_type

stmSetBitArrayLimitsSubLocalVar

Defines additional parameters for a “Bitarray” Subroutine local variable: left and right indices and default value. An error is issued if the data item is not of “Bitarray” type.

Syntax

```
stmSetBitArrayLimitsSubLocalVar (handle, chart_name, sub_name,
local_var_name, sub_implementation, bitarray_lindex, bitarray_rindex,
default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub _name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implementation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
bitarray_lindex	Input	String	Bit array's left bound
bitarray_rindex	Input	String	Bit array's right bound
default_val	Input	String	(*)Optional

stmSetBitLimits

Defines default value for a “Bit” element. An error is issued if the element is not of “Bit” type.

Syntax

```
stmSetBitLimits (handle, chart_name, dataitem_name, element_type,
default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
default_val	Input	String	Default value: either a numeric constant or a data item name

Note

This API supports all bit-type elements that may have a default value (except for field, subroutine parameter, and subroutine local variable):

- data_item
- data_type

stmSetBitLimitsField

Defines default value for “Bit” field. An error is issued if the field is not of “Bit” type.

Syntax

```
stmSetBitLimitsField (handle, chart_name, data_item_name, field_name,
element_type, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item /User-defined-type name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
default_val	Input	String	Default value: either a numeric constant or a data item name

Note

This API supports all element types that may have a “Bit” Field:

- data_item
- data_type

stmSetBitLimitsSubLocalVar

Defines default value for a “Bit” Subroutine local variable. An error is issued if the local variable is not of “Bit” type.

Syntax

```
stmSetBitLimitsSubLocalVar (handle, chart_name,  sub_name, local_var_name,
sub_implementation, min_val, max_val, num_of_bits, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implemntation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
default_val	Input	String	Default value: either a numeric constant or a data item name

stmSetConditionLimits

Defines default value for a “Condition.” An error is issued if the element is not of “Condition” type.

Syntax

```
stmSetConditionLimits (handle, chart_name, dataitem_name, element_type,
default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Condition name
element_type	Input	Enumeration stm_element_type	stm_condition stm_data_type (UserType)
default_val	Input	String	Default value: either a numeric constant or a data item name

Note

This API supports all condition type elements that may have a default value (except for field, subroutine parameter, and subroutine local variable):

- data_item
- data_type

stmSetConditionLimitsField

Defines default value for a “Condition” field. An error is issued if the field is not of “Condition” type.

Syntax

```
stmSetConditionLimitsField (handle, chart_name, dataitem_name, field_name,
element_type, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Condition name
field _name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	stm_condition stm_data_type (UserType)
default_val	Input	String	Default value: either a numeric constant or a data item name

Note

This API supports all element types that may have a “Condition” field:

- data-item
- data_type

stmSetConditionLimitsSubLocalVar

Defines default value for a “Condition” Subroutine local variable. An error is issued if the element is not of “Condition” type.

Syntax

```
stmSetConditionLimitsSubLocalVar (handle, chart_name, sub_name,
local_var_name, sub_implementation, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implementation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
default_val	Input	String	Default value: either a numeric constant or a data item name

stmSetEnumTypeLimits

Defines default value for an “enum” user-defined-type. An error is issued if the user-defined-type is not of “enum” type.

Syntax

```
stmSetEnumTypeLimits (handle, chart_name, dataitem_name, element_type,
default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_type (UserType)
default_val	Input	String	Default value is an enum value

stmSetHandleErrorFunc

Setsan error handler function.

Syntax

```
stmSetHandleErrorFunc(stm_context, error_func(error_message));
```

Return Type

```
void
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
Error_func	Input	Pointer to functios	Function to call in case of an error
Error_message	Input	String	String to pass to the error function

stmSetIntegerLimits

Defines additional parameters for an integer data item: # of bits, min/max values, and default value. An error is issued if the data item is not of integer type.

Syntax

```
stmSetIntegerLimits (handle, chart_name, dataitem_name, element_type,
min_val, max_val, num_of_bits, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
min_val	Input	String	Min value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
num_of_bits	Input	String	Number of bits for the integer value (*)Optional - a numeric constant
default_val	Input	String	(*)Optional

stmSetIntegerLimitsField

Defines additional parameters for an integer field: # of bits, min/max values, and default value. An error is issued if the data item is not of integer type.

Syntax

```
stmSetIntegerLimitsField (handle, chart_name, dataitem_name, field_name,
element_type, min_val, max_val, num_of_bits, default_val)
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem_name	Input	String	Data item/User-defined-type name
field_name	Input	String	Field Name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
min_val	Input	String	Min value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
num_of_bits	Input	String	Number of bits for the integer value (*)Optional - a numeric constant
default_val	Input	String	(*)Optional

Note

This API supports all element types that may have an Integer Field:

- data_item
- data_type

stmSetIntegerLimitsSubLocalVar

Defines additional parameters for an integer Subroutine local variable: # of bits, min/max values, and default value. An error is issued if the data item is not of integer type.

Syntax

```
stmSetIntegerLimitsSubLocalVar (handle, chart_name,  sub_name,
local_var_name, sub_implementation, min_val, max_val, num_of_bits,
default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implementation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
min_val	Input	String	Min value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
num_of_bits	Input	String	Number of bits for the integer value (*)Optional - a numeric constant
default_val	Input	String	(*)Optional

stmSetIntegerLimitsSubParameter

Defines additional parameters for an integer Subroutine parameter: # of bits and min/max values. An error is issued if the data item is not of integer type.

Syntax

```
stmSetIntegerLimitsSubParameter (handle, chart_name, sub_name, param_name,
min_val, max_val, num_of_bits)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
param_name	Input	String	Parameter name
min_val	Input	String	Min value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
num_of_bits	Input	String	Number of bits for the integer value (*)Optional - a numeric constant

stmSetRealLimits

Defines additional parameters for a “Real” type element: min/max values and default value. An error is issued if the data item is not of “Real” type.

Syntax

```
stmSetRealLimits (handle, chart_name, dataitem_name, element_type, min_val,
max_val, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem_name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
min_val	Input	String	Max value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
default_val	Input	String	Default value: (*)Optional either a numeric constant or a data item name

Note

This API supports all “Real” type elements (except for field, subroutine parameter, and subroutine local variable):

- data_item
- data_type

stmSetRealLimitsField

Defines additional parameters for a “Real” field: min/max values and default value. An error is issued if the field is not of “Real” type.

Syntax

```
stmSetRealLimitsField (handle, chart_name, dataitem_name, field_name,
element_type, min_val, max_val, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem_name	Input	String	Data item/User-defined-type name
field_name	Input	String	Field Name
element_name	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
min_val	Input	String	Min value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
default_val	Input	String	(*)Optional

Note

This API supports all element types that may have a “Real” Field:

- data_item
- data_type

stmSetRealLimitsSubLocalVar

Defines additional parameters for a “Real” Subroutine local variable: min/max values and default value. An error is issued if the local variable is not of “Real” type.

Syntax

```
stmSetRealLimitsSubLocalVar (handle, chart_name,  sub_name, local_var_name,
sub_implementation, min_val, max_val, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implementation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
min_val	Input	String	Max value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name
default_val	Input	String	(*)Optional

stmSetRealLimitsSubParameter

Defines additional parameters for a “Real” Subroutine parameter: min/max values. An error is issued if the parameter is not of “Real” type.

Syntax

```
stmSetRealLimitsSubParameter (handle, chart_name, sub_name, param_name,
min_val, max_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
param_name	Input	String	Parameter name
min_val	Input	String	Max value: either a numeric constant or a data item name
max_val	Input	String	Max value: either a numeric constant or a data item name

stmSetRouterMode

Defines the router transparency mode.

Syntax

```
stmSetRouterMode (STMContextHandle, chart_name, router_name,
tranparency_mode)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
STMContextHandle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
router_name	Input	String	Router name
transparency_mode	Input	stm_router_mode	stm_transparent_router/ stm_non_transparent_router

stmSetShortDescriptionSubParameters

Sets short-description for an existing parameter in a specified subroutine.

Syntax

```
stmSetShortDescriptionSubParameters (STMContextHandle, chart_name, sub_name,
param_name, new_short_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Arguemnt Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
param_name	Input	String	Parameter name
new_short_description	Input	String	(*)Optional Parameter short description

stmSetShortDescriptionSubLocalVar

Sets short-description for an existing local-variable in a specified subroutine.

Syntax

```
stmSetShortDescriptionSubLocalVar (STMContextHandle, chart_name, sub_name,
local_var_name, sub_implementation, new_short_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local-variable name
sub_implementation	Input	stm_sb_select_ implementation	Subroutine implementation for the local-variable
new_short_description	Input	String	(*)Optional Short-description for the local-variable

stmSetShortLongDescription

Sets short and long descriptions for an element.

Syntax

```
stmSetShortLongDescription(stm_context, chart_name, element_name,
element_type, new_short_description, new_long_description)
```

Return Type

```
status: success / fail
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
element_type	Input	Enumeration stm_element_type	Element type: stm_chart / stm_activity / stm_state / stm_data_item / stm_data_type (UserType) / stm_condition / stm_event / stm_action
new_short_description	Input	String	(*) Optional NULL means no change in current description Empty string will erase current description
new_long_description	Input	String	(*) Optional NULL means no change in current description Empty string will erase current description

stmSetShortLongDescriptionField

Sets short and long descriptions for a record/union field.

Syntax

```
stmSetShortLongDescriptionField (STMContextHandle, chart_name, element_name,
field_name, element_type, new_short_description, new_long_description)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Integer	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Element type of the field
new_short_description	Input	String	Subroutine implementation for the local-variable
new_long_description	Input	String	(*)Optional Long description for the field

stmSetStringLength

Defines additional parameters for a string data item: string length. An error is issued if the data item is not of string type.

Syntax

```
stmSetStringLength (stm_context, chart_name, dataitem_name, element_type,
string_length)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
string_length	Input	String	String length: either a numeric constant or a data item name

stmSetStringLengthField

Defines additional parameters for a string type field: string length. An error is issued if the field is not of string type.

Syntax

```
stmSetStringLengthField (STMContextHandle, chart_name, data_item_name,
field_name, element_type, string_length)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
data_item_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Field element type
string_length	Input	String	Value to set as String length

stmSetStringLengthSubLocalVar

Set the string length for a String type subroutine local-variable.

Syntax

```
stmSetStringLengthSubLocalVar (STMContextHandle, chart_name, sub_name,
local_var_name, sub_implentation, string_length)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
local_var_name	Input	String	Local-variable name
sub_implementation	Input	stm_sb_select_ implementation	Field element type
string_length	Input	String	Value to set as String length

stmSetStringLengthSubParameter

Set the string length for a String type subroutine parameter.

Syntax

```
stmSetStringLengthSubParamter (STMContextHandle, chart_name, sub_name,
param_name, string_length)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub_name	Input	String	Subroutine name
param_name	Input	String	Parameter name
string_length	Input	String	Value to set as string length

stmSetStringLimits

Defines additional parameters for a “String”element: string length and default value. An error is issued if the element is not of “String” type.

Syntax

```
stmSetStringLimits (handle, chart_name, dataitem_name, element_type,
string_length, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
string_length	Input	String	String length
default_val	Input	String	(*)Optional

Note

This API supports all “String” type elements (except for field, subroutine parameter, and subroutine local variable):

- data_item
- data_type

stmSetStringLimitsField

Defines additional parameters for a “String” Field: string length and default value. An error is issued if the field is not of “String” type.

Syntax

```
stmSetStringLimitsField (handle, chart_name, dataitem_name, field_name,
element_type, string_length, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
dataitem _name	Input	String	Data item/User-defined-type name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	stm_data_item stm_data_type (UserType)
string_length	Input	String	String length
default_val	Input	String	(*)Optional

Note

This API supports all element types that may have a “String” Field:

- data_item
- data_type

stmSetStringLimitsSubLocalVar

Defines additional parameters for a “String” Subroutine local variable: string length and default value. An error is issued if the local variable is not of “String” type.

Syntax

```
stmSetStringLimitsSubLocalVar (handle, chart_name,  sub_name, local_var_name,
sub_implementation, string_length, default_val)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
handle	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sub _name	Input	String	Subroutine name
local_var_name	Input	String	Local variable name
sub_implementation	Input	Enumeration stm_sb_select_ implementation	The referenced subroutine impelementation
string_length	Input	String	String length
default_val	Input	String	(*)Optional

stmSetSTStaticReaction

Sets static reaction for an existing state in a specified statechart. If a static reaction exists for this state, it is replaced.

Syntax

```
stmSetSTStaticReaction (stm_context, chart_name, state_name, static_reaction)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Statechart name
state_name	Input	String	State name
static_reaction	Input	String	Static reaction for the state

stmUserTypeChangeScope

Changes the definition scope of a given user defined type.

Syntax

```
stmUserTypeChangeScope (stm_context, chart_name, UserType_name, new_chart
_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
usertype_name	Input	String	User type name
new_chart _name	Input	String	New chart name

Delete Functions

This sections contains the following delete functions:

- stmDeleteAction
- stmDeleteActivity
- stmDeleteAllAttributes
- stmDeleteAllAttributesField
- stmDeleteAttribute
- stmDeleteAttributeField
- stmDeleteChartOrGDS
- stmDeleteCondition
- stmDeleteConnector
- stmDeleteDataFlow
- stmDeleteDataItem
- stmDeleteDefaultTransition
- stmDeleteEvent
- stmDeleteExternalActivity
- stmDeleteExternalRouter
- stmDeleteInfoFlow
- stmDeleteLifeline
- stmDeleteMessage
- stmDeletePartition
- stmDeleteRouter
- stmDeleteState

- **stmDeleteSubroutine**
- **stmDeleteTransition**
- **stmDeleteUserType**
- **stmRemoveInfoFlowComponent**
- **stmRemoveSubroutineImplementation**

stmDeleteAction

Deletes an event.

Syntax

```
stmDeleteAction (stm_context, chart name, action name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
action_name	Input	String	Action name

stmDeleteActivity

Deletes an activity.

Syntax

```
stmDeleteActivity (stm_context, chart name, unique activity name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
unique_activity_name	Input	String	Unique Activity name

stmDeleteAllAttributes

Deletes all attributes of a specified element.

Syntax

```
stmDeleteAllAttributes (STMContextHandle, chart_name, element_name,
element_type)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
element_type	Input	Enumeration stm_element_type	Element type

stmDeleteAllAttributesField

Deletes all attributes of a field.

Syntax

```
stmDeleteAllAttributesField (STMContextHandle, chart_name, element_name,
field_name, element_type)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Field element type

stmDeleteAttribute

Deletes a single attribute of a specified element.

Syntax

```
stmDeleteAttribute (STMContextHandle, chart_name, element_name, element_type,
attr_name, delete_enforced)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
element_type	Input	Enumeration stm_element_type	Element type
attr_name	Input	String	Attribute name
delete_enforced	Input	stm_boolean	If stm_true - delete attribute even if "enforced"

stmDeleteAttributeField

Deletes an attribute of a record/union field.

Syntax

```
stmDeleteAttributeField (STMContextHandle, chart_name, element_name,
field_name, element_type, attr_name, delete_enforced)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
element_name	Input	String	Element name
field_name	Input	String	Field name
element_type	Input	Enumeration stm_element_type	Field element type
attr_name	Input	String	Attribute name
delete_enforced	Input	stm_boolean	If stm_true - delete attribute even if “enforced”

stmDeleteChartOrGDS

Deletes a chart or a GDS.

Syntax

```
stmDeleteChartOrGDS(stm_context, chart_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name

stmDeleteCondition

Deletes a condition.

Syntax

```
stmDeleteCondition (stm_context, chart or_gds_name, condition name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart or_gds_name	Input	String	Chart or GDS name
condition_name	Input	String	Condition name

stmDeleteConnector

Deletes a connector from a chart.

Syntax

```
stmDeleteConnector(stm_context , chart_name, cn_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
cn_name	Input	String	Connector name

stmDeleteDataFlow

Deletes the dataflow from source_box_name to target_box_name carrying the given label. If more then one transition satisfies this, one arbitrary is deleted**.**

Syntax

```
stmDeleteDataFlow (stm_context, chart name, source_box_name, target_box_name, label)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
source_box_name	Input	String	Source_box_name
target_box_name	Input	String	Target_box_name
label	Input	String	Label (*)Optional

stmDeleteDataItem

Deletes a data item.

Syntax

```
stmDeleteDataItem (stm_context, chart or_gds_name, data_item name
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
Chart or_gds_name	Input	String	Chart or GDS name
Data_item_name	Input	String	Data item name

stmDeleteDefaultTransition

Deletes the default transition that goes to the to target_box_name carrying the given label. If more than one transition satisfies this, one arbitrary will be deleted**.**

Syntax

```
stmDeleteDefaultTransition (stm_context, chart name, target_box_name, label)
```

Return Type

`status_code`

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
target_box_name	Input	String	Target_box_name
label	Input	String	Label (*)Optional

stmDeleteEvent

Deletes an event.

Syntax

```
stmDeleteEvent (stm_context, chart name, event name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
event_name	Input	String	event name

stmDeleteExternalActivity

Deletes an external activity. Each occurrence is identified by the user key.

Syntax

```
stmDeleteExternalActivity (stm_context, chart name, external activity name,
user_key)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
external_activity_name	Input	String	Unique Activity name
user_key	Input	String	A user key should be a unique string to identify occurrences of an external activity

stmDeleteExternalRouter

Deletes an external router. This command deletes all occurrences of the box in the chart.

Syntax

```
stmDeleteExternalRouter (STMContextHandle, chart_name, router_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
router_name	Input	String	Router name

stmDeleteInfoFlow

Deletes an information flow.

Syntax

```
stmInfoFlow(stm_context, chart_name, info_flow_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Info_flow_name	Input	String	Information flow name

stmDeleteLifeline

Deletes a lifeline from a sequence diagram.

Syntax

```
stmDeleteLifeline(stm_context, chart_name, lifeline_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
lifeline_Item_name	Input	String	Lifeline name

stmDeleteMessage

Deletes a message from a sequence diagram.

Syntax

```
stmDeleteMessage(stm_context, chart_name, source_lifeline_name,
target_lifeline_name ,label, y);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
source_lifeline_Item_ name	Input	String	Source Lifeline name
Target_lifeline_Item_ name	Input	String	Target Lifeline name
label	Input	String	Message label
y	Input	Double	Y coordinate of message

stmDeletePartition

Deletes a partition line from a sequence diagram.

Syntax

```
stmDeletePartition(stm_context, chart_name, partition_name, y);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart nake
partition_name	Input	String	Partition line name
y	Input	Double	Y coordinate of Partition Line

stmDeleteRouter

Deletes a router.

Syntax

```
stmDeleteRouter (STMContextHandle, chart_name, router_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Name of the chart
router_name	Input	String	Name of the router

stmDeleteState

Deletes a state.

Syntax

```
stmDeleteState (stm_context, chart name, unique state name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
Chart name	Input	String	Chart name
Unique_state_name	Input	String	Unique state name

stmDeleteSubroutine

Deletes a subroutine.

Syntax

```
stmDeleteSubroutine (STMContextHandle, chart_name, sbr_name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name

stmDeleteTransition

Deletes the transition from source_box_name to target_box_name carrying the given label. If more than one transition satisfies this, one arbitrary is deleted**.**

Syntax

```
stmDeleteTransition (stm_context, chart name, source_box_name,
target_box_name, label)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart name	Input	String	Chart name
source_box_name	Input	String	Source_box_name
target_box_name	Input	String	Target_box_name
label	Input	String	Label (*)Optional

stmDeleteUserType

Deletes a user defined type.

Syntax

```
stmDeleteUserType (stm_context, chart or_gds_name, user_type name)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart or_gds_name	Input	String	Chart or GDS name
user_type_name	Input	String	User type name

stmRemoveInfoFlowComponent

Deletes a component from an information flow.

Syntax

```
stmRemoveInfoFlowComponent(stm_context, chart_name, info_flow_name,
component_name);
```

Return Type

```
status_code
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
Stm_context	Input	Pointer	STM context descriptor that was returned by StmInitImport
chart_name	Input	String	Chart name
Info_flow_name	Input	String	Information Flow name
Component_name	Input	String	Component name

stmRemoveSubroutineImplementation

Removes subroutine implementation.

Syntax

```
stmRemoveSubroutineImplementation (STMContextHandle, chart_name, sbr_name,
sbr_impl_type,)
```

Return Type

```
status_code
```

See **Function Status Codes** for the list of possible values.

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
chart_name	Input	String	Chart name
sbr_name	Input	String	Subroutine name
sbr_impl_type	Input	stm_sb_select_ implementation	Subroutine implementation to remove

General Purpose Functions

This section contains the following general purpose functions:

- **stmCloseImport**
- **stmInitImport**
- **stmSetHandleErrorFunc**
- **stmSetOffsetValuesOfBoxName**
- **stmSetStmHandleErrorFunc**

stmCloseImport

Closes a data import session. This call results in updating the workarea given by the previous **stmInitImport** with all the new data that was created between the two calls.

Syntax

```
stmCloseImport(stm_context)
```

Return Type

```
status_code: success / fail
```

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport

stmInitImport

Initializes a data import session.

Syntax

```
stmInitImport(workarea_dir, project_name)
```

Return Type

STMContextHandle: a pointer to a context structure. This pointer should be used as a context descriptor when calling the other functions.

If a failure occurs, a NULL pointer is returned (`workarea_dir` does not exist**)**

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
workarea_dir	Input	String	Full path to a Rational Statemate workarea, where the imported data will be created
project_name	Input	String	Rational Statemate project name

stmSetHandleErrorFunc

Sets an error handler function.

Syntax

```
stmSetHandleErrorFunc(stm_context, error_function_ptr(err_str))
```

Return Type

void

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
error_func	Input	Pointer to function	Function to call in case of an error.
error_message	Input	String	String to pass to error function.

stmSetOffsetValuesOfBoxName

Defines the offset values for the position of the box name.

(default values are : x_offset = 0.5, y_offset = -0.75).

Syntax

```
stmSetOffsetValuesOfBoxName (stm_context, x_offset, y_offset)
```

Return Type

void

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
x_offset	Input	Double	X offset value
y_offset	Input	Double	Y offset value

stmSetStmHandleErrorFunc

Defines the error function for displaying the interface error messages.

Syntax

```
stmSetstmHandleErrorFunc (STMContextHandle, handle, error_func, msg_str)
```

Return Type

void

Arguments

Argument Name	Input/ Output	Argument Type	Argument Description
stm_context	Input	Pointer	STM context descriptor that was returned by stmInitImport
error_func	Input	Pointer	Pointer to a function with one string argument (error message) and one integer argument (error severity): void (*error_func)(char* msg_str,int severity)

Function Status Codes

Data Import functions return the function status code. This code reports whether the function call was successfully completed. If the function call fails, the status code indicates the problem. This status code can be used to pinpoint run-time errors in your program.

The status code is an integer value. The Data Import provides predefined constants for the function status codes. This enables you to use the status name attached to each status code in your program.

Status Code Severity

Status codes have three severity levels:

- **S** for success
- **W** for warning
- **E** for error

You should check the return status codes to ensure that your function call is successful.

Status Code Definitions

The following table lists the status codes, names, definitions, and their severity levels.

Code	Status Name	Severity Level
–4	`stm_no_stm_root` **UNIX**: The `STM_ROOT` environment variable is not defined. **VMS**: The `STM$ROOT` or `STM$PM` logical name does not exist.	E
–3	`stm_obsolete_function` Irrelevant function for the current version.	E
–2	`stm_missing_elements_in_list` Input elements do not exist in the database.	W

–1	`stm_list_type_mismatch` Incorrect element type used in the query.	E
0	`stm_success` The function call was successful.	S
1	`stm_id_out_of_range` The specified ID is not valid for this element type.	E
2	`stm_id_not_found` An element with the specified ID does not exist.	E
3	`stm_illegal_name` The specified name is not legal.	E
4	`stm_name_not_found` The specified name does not exist.	E
5	`stm_name_not_unique` There is more than one element with the specified name, so a specific path name is required.	E
6	`stm_missing_name` The specified element has no name.	W
7	`stm_missing_synonym` The specified element has no synonym.	W
8	`stm_missing_short_description` The specified element has no short description.	W
9	`stm_missing_long_description` The specified element has no long description.	W
10	`stm_attribute_name_not_found` The specified element has no attribute name.	W
11	`stm_starting_keyword_not_found` The long description of the specified element does not contain the given starting keyword.	W
12	`stm_ending_keyword_not_found` The long description of the specified element does not contain the given ending keyword.	W
13	`stm_primitive_element` The element is primitive.	W
14	`stm_can_not_open_file` The operating system cannot open the file with the specified name.	E

15	`stm_illegal_address` The pointer address is illegal.	E
16	`stm_not_an_and_state` This state is not supposed to contain and-lines.	W
17	`stm_no_and_lines_in_and_state` This and-state is missing and-lines.	E
18	`stm_missing_graphic_data` Graphic data is missing from the element.	E
19	`stm_nil_list` There is no input list.	E
20	`stm_list_element_does_not_exist` The specified element does not exist.	W
21	`stm_cannot_compare_lists` The lists cannot be compared because the list types are different, or the lists are not initialized.	E
22	`stm_elements_without_name` The list cannot be sorted because its elements have no names.	E
23	`stm_elements_without_synonym` The list cannot be sorted because its elements have no synonyms.	E
24	`stm_elements_not_hierarchical` The list cannot be sorted because it is not hierarchical.	E
25	`stm_illegal_extract_type` You cannot extract this element type.	E
26	`stm_no_such_list` No such list exists	E
27	`stm_not_diagram_connector` There is no value in a connector that is not a diagram connector.	E
28	`stm_implicit_element` The element is defined implicitly—it is an internally defined entity.	E
29	`stm_missing_label` The element has no label.	W
30	`stm_unknown_plotter` The plotter type is unknown.	E

31	`stm_unresolved` The element is unresolved.	W
32	`stm_elements_without_attributes` The list cannot be sorted because its elements have no attributes.	E
33	`stm_not_instance` The element is not an instance.	E
34	`stm_no_updated_pmdb` The workarea database is not updated to the current version.	E
35	`stm_no_updated_projdb` The installation database is not updated to the current version.	E
36	`stm_no_legal_operator` The user is not authorized as a Rational Statemate operator.	E
37	`stm_deadlock` Deadlock situation.	E
38	`stm_not_member_of_project` The user is not a member of the specified project.	E
39	`stm_nonexistent_project` The specified project does not exist.	E
40	`stm_not_enough_memory` The plot cannot be produced because there is not enough memory.	E
41	`stm_empty_chart` The plot file cannot be produced because the chart is empty.	E
42	`stm_plot_failure` The plot file was not produced because of a system error.	E
43	`stm_no_file_of_licensed_host` The file containing the name of the licensed host does not exist.	E
44	`stm_empty_file_of_licensed_host` The file containing the name of the licensed host is empty.	E
45	`stm_cannot_chdir_to_work_area` Could not change directory to the workarea.	E
46	`stm_cannot_write_to_file` No space is left on device for writing a file.	E

47	`stm_illegal_parameter` An illegal parameter value was supplied.	E
48	`stm_illegal_parameter_mode` Illegal parameter mode.	E
49	`stm_illegal_parameter_name` Illegal parameter name.	E
50	`stm_null_string` The input string is null.	E
51	`stm_illegal_len` The length value is illegal.	E
52	`stm_illegal_index` The index value is illegal.	E
53	`stm_cannot_read_file` Cannot read from a file that was not opened.	E
54	`stm_end_of_file` Reached the end-of-file.	E
55	`stm_not_a_parameter` The specified ID is not a parameter.	E
56	`stm_param_not_compatible` The actual and formal parameters are not compatible.	W
57	`stm_error_in_file` There is an error in the requirement file.	E
58	`stm_missing_field` A field is missing in the requirement record.	W
59	`stm_missing_user_type` The specified element has no user-defined type.	E
60	`stm_illegal_attribute_name` The attribute name is illegal.	E
61	`stm_illegal_attribute _value` The attribute value is too long.	E
62	`stm_duplicate_attribute _pair` The specified attribute name/value pair already exists.	E

63	`stm_not_in_rw_transaction` Attempt to modify the database when not in a read/write transaction.	E
64	`stm_missing_of_enum_type` The specified element has no enumerated type associated with its array type definition.	W
65	`stm_missing_user_code` The specified element has no user code.	W
66	`stm_missing_subroutine_params` The specified element has no subroutine parameters.	W
67	`stm_missing_local_data` The specified element has no local data.	W
68	`stm_missing_global_data` The specified element has no global data.	W
69	`stm_no_connected_chart` The specified element is not connected to a chart.	W
70	`stm_attribute_cannot_be_deleted` The specified element's attribute cannot be deleted.	E
71	`stm_missing_cbk_binding` The specified element has no callback binding.	W
72	`stm_missing_subroutine_binding` The specified element has no subroutine binding.	W
73	`stm_missing_statemate_action_lang` The specified element has no action language.	W
74	`stm_no_projects` There are no projects in the project management database.	W
75	`stm_member_has_no_wa`	E
76	`stm_missing_external_link` Specific element has no long description.	W
77	`stm_not_chart_id`	E
78	`stm_message_not_found`	E
79	`stm_not_referenced_sd`	E
80	`stm_not_timing_constraint`	E

81	stm_not_order_insignificant	E
82	stm_missing_note Element has no note.	W
83	stm_missing_description_file Element does not have an external description file defined.	W
84	stm_not_boundry_box	E
85	stm_not_use_case	E
86	stm_not_actor	E
87	stm_not_partition	E
88	stm_not_sequence_diagram	E
89	stm_not_activity	E
90	stm_invalid_use_case_scen_num	E
91	stm_missing_extention_point_definition	W
92	stm_missing_timing_constraint_note	W
93	stm_no_use_case_scen_attr_defined	W
94	stm_use_case_scen_attr_val_not_defined	W
95	stm_illegal_chart	E
96	stm_info_flow_component_exists	E
97	stm_missing_info_flow_component	W
98	stm_generic_chart_not_in_database	E
99	stm_cannot_delete_parent_of_control_activity	E
100	stm_hyperlinked_expression_not_implemented_for_plotter	W
101	stm_illegal_param_min_val	E
102	stm_illegal_param_max_val	E
103	stm_illegal_param_ba_lindex	E
104	stm_illegal_param_ba_rindex	E
105	stm_illegal_param_user_type	E

106	stm_illegal_param_enum_type	E
107	stm_illegal_param_type	E
108	stm_illegal_param_structure_type	E
109	stm_illegal_local_var_structure_type	E
110	stm_illegal_short_description_length	E
111	stm_illegal_local_var_min_val	E
112	stm_illegal_local_var_max_va	E
113	stm_illegal_local_var_ba_lindex	E
114	stm_illegal_local_var_ba_rindex	E
115	stm_illegal_local_var_user_type	E
116	stm_implementation_missing	E
117	stm_implementation_exists	E
118	stm_missing_subroutine	E
119	stm_illegal_global_var_mode	E
120	stm_illegal_global_var_name	E
121	stm_illegal_expression_n_chart There is an illegal expression in the loaded chart.	E
122	stm_error_in_chart There is an error in the loaded chart.	E
123	stm_cannot_open_chart_file Cannot open the chart file to be loaded.	E
124	stm_exceeded_max_id_number There are more than 1023 IDs in the workarea.	E
125	stm_chart_not_in_database Cannot find a chart in the database to be saved or unloaded.	E
126	stm_file_not_in_work_area Cannot find a file in the workarea to be saved or unloaded.	E
127	stm_cannot_copy_file Cannot copy a file during a save or load operation.	E

128	`stm_cannot_create_file` Cannot create an auxiliary file during a load to the workarea.	E
129	`stm_illegal_version` An illegal version was specified for the load operation.	E
130	`stm_file_not_found` Cannot find a source file in the load operation.	E
131	`stm_not_loaded_because_modified` A modified version of loaded chart or file exists in the workarea.	E
132	`stm_not_loaded_because_new` A new version of the loaded chart or file exists in the workarea.	E
133	`stm_not_unloaded_modified` The chart or file to be unloaded is modified.	E
134	`stm_not_unloaded_new` The chart or file to be unloaded is new.	E
135	`stm_chart_is_active` The chart to be unloaded is currently being edited by a graphics editor.	E
136	`stm_error_in_save_operation` There was a write to disk error during the save operation.	E
137	`stm_illegal_load_mode` An illegal mode was specified for the load operation.	E
138	`stm_not_loaded_because_type` A chart with the same name, but of another type, exists in the workarea.	E
139	`stm_illegal_type` An illegal type of configuration item was specified.	E
140	`stm_illegal_parameters` An illegal parameter to the load function was specified.	E
141	`stm_illegal_bindings` There is an error in the loaded chart file.	E
142	`stm_too_long_line` There is a line too long in the loaded chart file.	E
143	`stm_instance_type_conflict` There is an instance type conflict in the loaded chart file.	E

144	stm_usage_conflict There is a usage conflict in the loaded chart file.	E
145	stm_unrecognized_format The loaded chart file contains an unrecognized conflict.	E
146	stm_double_chart_parameters There is an error in the loaded chart file.	E
147	stm_double_chart_bindings There is an error in the loaded chart file.	E
148	stm_no_bindings	W
149	stm_missing_truth_table	E
150	stm_truth_table_invalid_row	E
151	stm_component_interface_changed	E
152	stm_cannot_load_component	E
153	stm_cannot_open_new_wa	E
154	stm_element_exists	E
156	stm_coordinates_out_of_range	E
157	stm_illegal_coordinates	E
158	stm_illegal_local_var_enum_type	E
159	stm_illegal_local_var_type	E
160	stm_illegal_local_var_name	E
161	stm_truth_table_invalid_column	E
162	stm_invalid_truth_table_cell	E
163	stm_truth_table_convert_failed	E
164	stm_conflicting_array_indices_types	E
165	stm_invalid_sd_scope	W
166	stm_sd_scope_not_defined	W
167	stm_use_all_public_gds	E
168	stm_error_in_backup	E

169	stm_not_message	E
170	stm_cannot_delete_file	E
171	stm_plot_illegal_option_key	E
172	stm_plot_illegal_option_val	E
173	stm_no_local_vars_in_selected_impelentation	E
174	stm_illegal_hyperlink_format	E
175	stm_illegal_font_name	E
176	stm_illegal_factor_value	E
177	stm_invalid_key	E
178	stm_no_legal_wa_operator The user is not authorized as the workarea operator.	E

Index

T

U

W