cādence™

Virtuoso© ADE Verifier SKILL Reference

Product Version IC23.1
November 2023

Contents

1

Introduction to Virtuoso ADE Verifier SKILL Functions

This manual describes the SKILL functions of Cadence© Virtuoso© ADE Verifier (Verifier).

Note: Only the functions and arguments described in this topic are supported for public use. Any undocumented functions or arguments are likely to be private and could be subject to change without notice. It is recommended that you check with your Cadence representative before using them.

This SKILL API reference is meant for verification project managers and designers who want to use Verifier SKILL APIs for requirements-based verification of their analog designs.

This topic assumes that users are familiar with the Cadence SKILL language and Virtuoso Verifier.

Licensing Requirements

Verifier is available with the Virtuoso_ADE_Verifier license (95270) in the ICADV12.3 and higher releases.

Appropriate licenses are required to simulate implementations from Verifier. These include the licenses for the following:

- Virtuoso ADE Assembler
- Virtuoso ADE Explorer
- A supported simulator for simulating the designs

Related Topics

Requirement Functions

Implementation Functions

1

Requirement Functions

Requirements form the hierarchical verification plan of your design verification project. You map requirements to their implementations. Virtuoso ADE Verifier (Verifier) determines the verification status of the requirements using this mapping information, along with other settings and results.

The SKILL functions that let you manage requirements in the ADE Verifier environment can be broadly classified in the following categories.

- Addition and Deletion Functions - These functions let you add and delete requirements.
  - verifAddReq
  - verifMoveReq
  - verifRemoveReq
- Manual Signoff Functions - These functions let you manually sign off requirements.
  - verifDeleteReqSignoff
  - verifGetReqSignoff
  - verifSignOffReq
- Data Extraction Functions - These functions let you extract data from requirements.
  - verifGetReferencedCellViews
  - verifGetReqParent
  - verifGetReqs
  - verifGetReqProp
  - verifGetReqProps
  - verifGetReqStatus
- Requirement Setup Functions - These functions let you set requirements.

    - verifCreateRandomId
    - verifSetReqCellviewHolder
    - verifSetReqId
    - verifSetReqProp
    - verifSetReqTitle
    - verifSetReqType

- Export and Import Functions - These functions let you export and import requirements.
    - verifCompareImportedFiles
    - verifExportJson
    - verifExportReqsToFile
    - verifGetImportedFiles
    - verifImportFile
    - verifMergeImportedFiles

- Custom Field Functions - These functions let you work with custom fields in requirements.
    - verifGetCustomFieldNames
    - verifGetCustomFieldValue
    - verifGetReqCustomFieldNames
    - verifGetReqCustomFieldValue
    - verifSetCustomFieldValue
    - verifSetReqCustomFieldValue

- Requirements Description Editor Functions - These functions let you edit the descriptions of requirements.
    - verifAddDocument
    - verifAddImage
    - verifDocumentExists
    - verifGetDocuments

- verifGetImages
- verifImageExists
- verifRemoveDocument
- verifRemoveImage

- Setup Library Functions - These functions let you manage the setup library cellviews.
  - verifGetSetupLibrary
  - verifSetSetupLibrary

Related Topics

Implementation Functions

Requirement-to-Implementation Mapping Functions

Verifier Session and Setup Functions

verifAddDocument

```
verifAddDocument(
     g_sessionId
     t_fileName
     )
     => t_name / nil
```

Description

Adds a file to the `documents` directory of the Verifier cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	The name of the file.

Value Returned

`t_name`	Relative location of the file in the Verifier cellview.
`nil`	The file could not be added or the operation is unsuccessful.

Examples

Opens a Verifier cellview and adds a file from the current directory to the session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.txt` from the current directory to the session.

```
verifAddDocument(sess "file.txt")
=> "documents/file.txt"
```

Related Topics

verifAddImage

verifDocumentExists

verifGetDocuments

verifAddImage

```
verifAddImage(
     g_sessionId
     t_fileName
     )
     => t_name / nil
```

Description

Adds a file to the `images` directory of the Verifier cellview.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the file.

Value Returned

t_name	Relative location of the file in the Verifier cellview.
`nil`	The file cannot be added or the operation is unsuccessful.

Examples

Opens a Verifier cellview and adds an image file to the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.png` from the current directory to the session.

```
verifAddDocument(sess "file.png")
=> "images/file.png"
```

Related Topics

verifAddDocument

verifDocumentExists

verifGetDocuments

verifGetImages

verifImageExists

verifRemoveDocument

verifRemoveImage

verifAddReq

```
verifAddReq(
     g_sessionId
     t_title
     [ ?reqId t_reqId ]
     [ ?parentId t_parentId ]
     [ ?pos x_pos ]
     )
     => t / nil
```

Description

Adds a new requirement with the title to the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_title	Title of the requirement.
`?reqId` *t_reqId*	Unique ID of the new requirement.
`?parentId` *t_parentId*	ID of an existing requirement that is the parent of the new requirement.
`?pos` *x_pos*	Position where the requirement will be inserted. By default, the new requirement is added at the end.

Value Returned

`t`	Requirement is added to the Verifier session.
`nil`	Requirement is not added to the Verifier session.

Examples

The following example opens a Verifier cellview and adds a requirement:

```
sess = verifOpenCellView("test" "sample" "verifier")
=>0
```

Creates a requirement with the given title.

```
verifAddReq(sess "full_diff_opamp")
=>t
```

Creates a requirement with ID `C1` under requirement `ID1`.

```
verifAddReq(sess "A child req" ?reqId "C1" ?parentId "ID1")
=>t
```

Related Topics

verifMoveReq

verifRemoveReq

verifCompareImportedFiles

```
verifCompareImportedFiles(
     g_sessionId
     l_files
     )
     => t / nil
```

Description

Compares the requirements from the Verifier session with the requirements from the imported files.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
l_files	The list of files to be compared. If no files are specified, all imported files are compared.

Value Returned

`t`	The comparison is successful and the differences are displayed.
`nil`	There specified imported files do not exist in the Verifier session or the command is unsuccessful.

Examples

The following example opens a Verifier cellview and compares the requirements in the session with the requirements imported from a CSV file:

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
```

```
verifCompareImportedFile(sessionId "test.csv" ?fileType "CSV" ?headerRows 1
?ignoreInvalidRows t)
=> t
```

Related Topics

verifExportJson

verifExportReqsToFile

verifGetImportedFiles

verifImportFile

verifMergeImportedFiles

verifCreateRandomId

```
verifCreateRandomId(
     [ ?idLength x_idLength ]
     [ ?prefix t_prefix ]
     [ ?suffix t_suffix ]
     )
     => t_id
```

Description

Creates a string of random characters that can be used as a unique requirement ID.

Arguments

`?idLength` *`x_idLength`*	The length of the generated portion of the ID. This does not include the given prefix or the suffix. Default is 5.
`?prefix` *`t_prefix`*	The prefix of generated IDs.
`?suffix` *`t_suffix`*	The suffix of generated IDs.

Value Returned

`t_id`	Generated random ID.

Examples

In your `.cdsinit` file, specify the following:

```
envSetVal("verifier.requirement" "idCreationFunction" 'string
"verifCreateRandomId")
```

Newly created requirements will have an ID of five random characters, such as NUX8L or pWnT8.

Opens a Verifier cellview and creates a random ID for a new requirement.

```
verifCreateRandomId()
=> "pWnT8"
```

Opens a Verifier cellview and creates a random ID for a new requirement with the prefix "`id`" and a suffix "`_x`".

```
verifCreateRandomId(?prefix "id_" ?suffix "_x")
=> "id_hhVGj_x"
```

Related Topics

verifSetReqCellviewHolder

verifSetReqId

verifSetReqProp

verifSetReqTitle

verifSetReqType

verifDeleteReqSignoff

```
verifDeleteReqSignoff(
     g_sessionId
     [ ?reqId g_reqId ]
     )
     => t / nil
```

Description

Deletes the signoff setup of the specified requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?reqId` *g_reqId*	Single requirement ID or list of requirement IDs.

Value Returned

`t`	Signoff setup of the specified requirement is deleted.
`nil`	There is no signoff information to delete, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves a requirement that has the verification type manual. The example then retrieves the sign-off details of that requirement (`ID1`). Then, it removes the sign-off and checks its removal.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 15:25:59 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffComments
"Manully passing this requirement until next run."
)
```

Deletes a signoff setup for a single requirement.

```
verifDeleteReqSignoff(uid ?reqId "ID1")
=> t
verifGetReqSignoff(uid "ID1")
=> nil
```

Deletes signoff setup for a list of requirements.

```
verifDeleteReqSignoff(uid ?reqId list("ID1" "ID2"))
=> t
```

Deletes signoff setup for all requirements.

```
verifDeleteReqSignoff(uid)
=> t
```

Related Topics

verifGetReqSignoff

verifSignOffReq

verifDocumentExists

```
verifDocumentExists(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Checks if the given file exists in the `documents` directory of the Verifier cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	The name of the file.

Value Returned

`t`	The file exists in the `documents` directory of the Verifier cellview
`nil`	The file does not exist.

Examples

Opens a Verifier cellview and checks if a specified document file exists in the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.txt` from the current directory to the session.

```
verifAddDocument(sess "file.txt")
=> "documents/file.txt"
```

Finds the file successfully.

```
verifDocumentExists(sess "file.txt")
=> t
```

Returns `nil` if the file is not found.

```
verifDocumentExists(sess "another_file.txt")
=> nil
```

Related Topics

verifAddDocument

verifAddImage

verifGetDocuments

verifGetImages

verifImageExists

verifRemoveDocument

verifRemoveImage

verifExportJson

```
verifExportJson(
     g_sessionId
     [ ?filename t_filename ]
     [ ?compact g_compact ]
     )
     => t_filename / nil
```

Description

Exports the Verifier session as a `json` file. The `json` file contains details about the session, including the requirements, implementations, mappings, and simulation results.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?filename` *`t_filename`*	Name of the file to export to. If no filename is given, the details are exported to a file in the format `<lib>_<cell>_<view>.json`.
`?compact` *`g_compact`*	Exports the `json` file in compact format if `nil`. Otherwise, it is indented and printed over multiple lines.

Value Returned

`t_filename`	Name of the file that was created.
`nil`	The command is unsuccessful or an error occurred.

Examples

The following example shows how to export the Verifier session information as a `json` file:

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0

verifExportJson(sess)
=> "/home/user/dir1/test_sample_verifier.json"

verifExportJson(sess ?filename "v1.json")
=> "/home/user/dir1/v1.json"

verifExportJson(sess ?filename "v2.json" ?compact t)
=> "/home/user/dir1/v2.json"
```

Related Topics

verifCompareImportedFiles

verifExportReqsToFile

verifGetImportedFiles

verifImportFile

verifMergeImportedFiles

verifExportReqsToFile

```
verifExportReqsToFile(
     g_sessionId
     t_fileName
     [ ?fileType t_fileType ]
     [ ?fields t_fields ]
     )
     => t / nil
```

Description

Exports the requirements to either an Excel or CSV file. It can also export the results to an imported file.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the file to export to.
`?fileType` *t_file-type*	A case insensitive string from: CSV, Excel, or Imported. The default is "`CSV`".
`?fields` *t_fields*	The list of requirement fields to be exported. These fields can contain the following case-insensitive names, listed in the default sequence: Overall Status, Mapping, History, Min Value, Max Value, Result Data Age, Passed, Failed, Not Run, Disabled. The requirement is always exported by default and cannot be disabled.

Value Returned

`t`	Successfully exported the requirements.
`nil`	Failed to export the requirements.

Examples

The following example shows how to use this function to export requirements:

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 1
```

```
verifExportReqsToFile(sess "reqs.csv")
=> t
verifExportReqsToFile(sess "reqs.xlsx" ?fileType "Excel")
=> t
verifExportReqsToFile(sess "reqs2.csv" ?fields "Overall Status,Mapping,Unmapped")
=> t
```

Related Topics

verifCompareImportedFiles

verifExportJson

verifGetImportedFiles

verifImportFile

verifMergeImportedFiles

verifGetCustomFieldNames

```
verifGetCustomFieldNames(
     g_sessionId
     )
     => l_fieldList / nil
```

Description

Retrieves a list of custom field names for a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_fieldList	List of the custom fields available to the specified Verifier session.
`nil`	There are no custom fields in the specified Verifier session, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves the custom fields available to the session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetCustomFieldNames(uid)
=> ("Email" "VerificationManager")
```

Related Topics

verifGetCustomFieldValue

verifGetReqCustomFieldNames

verifGetReqCustomFieldValue

verifGetCustomFieldValue

```
verifGetCustomFieldValue(
     g_sessionId
     t_fieldName
     )
     => l_fieldvalue / nil
```

Description

Retrieves the value of a custom field available in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fieldName	String specifying the name of the custom field.

Value Returned

l_fieldvalue	List of values for the specified custom field in the session.
`nil`	There is no custom field in the session, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves the value of the custom field:

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetCustomFieldValue(uid "Email")
=> tom@cadence.com"
```

Related Topics

verifGetCustomFieldNames

verifGetReqCustomFieldNames

verifGetReqCustomFieldValue

verifSetCustomFieldValue

verifSetReqCustomFieldValue

verifGetDocuments

```
verifGetDocuments(
     g_sessionId
     [ g_relative ]
     )
     => l_names / nil
```

Description

Returns the list of files that have been added to the `documents` directory of the Verifier cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2).`
`g_relative`	Determines whether to add the relative location of the file within the cellview to the document or not. If it is `nil` then the relative location of the file is not added to the list of file names.

Value Returned

`l_names`	List of files saved in the `documents` directory of the Verifier cell-view.
`nil`	The list of files does not exist or cannot be retrieved.

Examples

Opens a Verifier cellview and retrieves the list of document files added to the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Gets the documents. In this case, there are none.

```
verifGetDocuments(sess)
=> nil
```

Adds `file.txt` from the current directory to the session.

```
verifAddDocument(sess "file.txt")
=> "documents/file.txt"
```

Adds `spec.pdf` from the current directory to the session and returns the list of files added to the `documents` directory.

```
verifAddDocument(sess "spec.pdf")
=> "documents/spec.pdf"
```

Related Topics

verifAddDocument

verifAddImage

verifDocumentExists

verifGetImages

verifImageExists

verifRemoveDocument

verifRemoveImage

verifGetImages

```
verifGetImages(
     g_sessionId
     [ g_relative ]
     )
     => l_names / nil
```

Description

Returns the list of image files that have been added to `images` directory of the Verifier cellview.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_relative	Determines whether to add the relative location of the file within the cellview to the image or not. If it is `nil` then the relative location of the image file is not added to the list of image file names.

Value Returned

l_names	List of image file names in the Verifier cellview.
`nil`	Image files do not exist or the operation is unsuccessful.

Examples

Opens a Verifier cellview and retrieves the list of image files from the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
verifGetImages(sess)
=> nil
```

Gets the images. In the current example, there are none.

```
verifAddImage(sess "file.png")
=> "images/file.png"
```

Adds `file.txt` from the current directory to the session.

```
verifAddDocument(sess "wave.gif")
=> "images/spec.pdf"
```

Gets the list of images.

```
verifGetImages(sess)
=> ("file.png" "wave.gif")
```

Gets the list of images with their relative locations within the Verifier cellview.

```
verifGetImages(sess t)
=> ("images/file.png" "images/wave.gif")
```

Related Topics

verifAddDocument

verifAddImage

verifDocumentExists

verifGetDocuments

verifImageExists

verifRemoveDocument

verifRemoveImage

verifGetImportedFiles

```
verifGetImportedFiles(
     g_sessionId
     )
     => l_importedFiles / nil
```

Description

Returns a list of all imported files that are associated with the specified session. If no imported files have been defined, `nil` is returned.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_importedFiles	List of all imported files that are associated with the specified session.
`nil`	No imported files are defined in the specified session.

Examples

The following example shows how the function can be used:

```
sessionId = verifOpenCellView("test" "export" "verifier")
=> 0
verifGetImportedFiles(sessionId)
=> ("./test_export.csv")
```

Related Topics

verifCompareImportedFiles

verifExportJson

verifExportReqsToFile

verifImportFile

verifMergeImportedFiles

verifGetReferencedCellViews

```
verifGetReferencedCellViews(
     g_sessionId
     )
     => l_referencedCellViews / nil
```

Description

Returns a list of all the referenced Verifier cellviews that can be associated with the specified session. If no referenced Verifier cellviews are defined, `nil` is returned.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_referencedCellViews	List of all referenced Verifier cellviews that can be associated with the specified session.
`nil`	No referenced Verifier cellviews have been defined.

Examples

The following example opens a Verifier cellview and retrieves the referenced Verifier cellviews from the specified session:

```
uid = verifOpenCellView("test" "ref" "verifier")
0
verifGetReferencedCellViews(uid)
(("test" "ref" "verifier_OpAmp")
("test" "ref" "verifier_amsPLL")
)
```

Related Topics

verifGetReqs

verifGetReqParent

verifGetReqProp

verifGetReqProps

verifGetReqStatus

verifGetReqCustomFieldNames

```
verifGetReqCustomFieldNames(
     g_sessionId
     )
     => l_fieldList / nil
```

Description

Retrieves the list of the custom fields available to the requirements in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_fieldList	List of the custom fields available to the requirements in the Verifier session.
`nil`	There are no custom fields available to the requirements in the Verifier session, or the command failed.

Examples

The following example opens a Verifier session and retrieves the names of the custom field:

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetCustomFieldNames(uid)
=> ("Email" "VerificationManager")
```

Related Topics

verifGetCustomFieldNames

verifGetCustomFieldValue

verifGetReqCustomFieldValue

verifSetCustomFieldValue

verifSetReqCustomFieldValue

verifGetReqCustomFieldValue

```
verifGetReqCustomFieldValue(
     g_sessionId
     t_reqId
     t_fieldName
     )
     => l_fieldValue / nil
```

Description

Retrieves the value of the specified custom field of a requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.
t_fieldName	String name of the custom field for requirements.

Value Returned

l_fieldValue	List of values of the specified custom field of the requirement.
`nil`	Specified custom field of the requirement does not exist, has no value, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves the value of the custom field for a requirement.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetReqCustomFieldNames(uid)
=> ("Email" "Engineer")
```

Returns `""` if the custom field is empty and contains no value.

```
verifGetReqCustomFieldValue(uid "ID1" "Email")
=> ""
```

Returns a value if the custom field contains the specified value.

```
verifGetReqCustomFieldValue(uid "ID1" "Engineer")
=> "Fred"
```

Returns `nil` if the query is for a non-existing custom field name.

```
verifGetReqCustomFieldValue(uid "ID1" "Project")
*WARNING* (VERIFIER-1239): unknown custom property name 'Project' on requirement
'ID1'
=> nil
```

Related Topics

verifGetCustomFieldNames

verifGetCustomFieldValue

verifGetReqCustomFieldNames

verifSetCustomFieldValue

verifSetReqCustomFieldValue

verifGetReqParent

```
verifGetReqParent(
     g_sessionId
     t_reqId
     )
     => t_parentReqId / nil
```

Description

Returns the ID of the parent of the specified requirement.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.

Value Returned

t_parentReqId	ID of the parent of the specified requirement ID.
`nil`	ID of the parent of the specified requirement ID is not returned.

Examples

The following example opens a Verifier cellview and gets the ID for the parent of the specified requirement.

```
sessionID=verifOpenCellView("test" "setup" "verifier")
=> 0
```

```
verifGetReqParent(sessionId "ID_1.1")
=> "ID_1"
```

Related Topics

verifGetReferencedCellViews

verifGetReqs

verifGetReqProp

verifGetReqProps

verifGetReqStatus

verifGetReqProp

```
verifGetReqProp(
     g_sessionId
     t_reqId
     t_propName
     [ g_inherited ]

     )
     =>t_propValue / nil
```

Description

Returns the value of the specified property of a requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.
t_propName	Name of the requirement property.The valid property names are `Parent`, `HierId`, `ID`, `Title`, `Type`, `MinSpec`, `MaxSpec`, `Unit`, `Owner`, `ReadOnly`, `Description`, and `VerificationSpace`.
g_inherited	Boolean value that controls the retrieval of property values. Setting it to `t` retrieves the inherited property value, whereas setting it to `nil` or a blank retrieves its own property value. This argument is optional. The default is `nil`.

Value Returned

t_propValue	Value of the specified property of a requirement in a Verifier session.
`nil`	Specified property of the requirement was not found.

Examples

The following example opens a Verifier cellview and retrieves the inherited `MaxSpec` value of a requirement because it does not have its own `MaxSpec`, but its parent has the `MaxSpec` value:

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
verifGetReqProp(1 "ID_1" "MaxSpec" t)
=> "9.98"
```

Related Topics

verifGetReferencedCellViews

verifGetReqs

verifGetReqParent

verifGetReqProps

verifGetReqStatus

verifGetReqProps

```
verifGetReqProps(
     g_sessionId
     t_reqId
     [ g_inherited ]
     )
     =>o_reqProps / nil
```

Description

Returns all the properties of a requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.
g_inherited	Boolean value that controls the retrieval of property values. Setting it to `t` retrieves the inherited property value, whereas setting it to `nil` or a blank retrieves its own property value. This argument is optional.

Value Returned

o_reqProps	Properties of a requirement in a Verifier session.
`nil`	Properties of a requirement in a Verifier session were not found.

Examples

The following example opens a Verifier cellview and retrieves the properties for the requirement `ID_1.1`:

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
tableToList(verifGetReqProps(sessionId "ID_1.1" t))
(("Unit" "")
("Description" "")
("MinSpec" "0.98")
("ID" "ID_1.1")
("Parent" "ID_1")
("MaxSpec" "")
("Title" "Swing")
("Type" "Spec Pass")
("Owner" "Tom")
("ReadOnly" "false")
("VerificationSpace" "")
("HierId" "1.1")
)
```

Related Topics

verifGetReferencedCellViews

verifGetReqs

verifGetReqParent

verifGetReqProp

verifGetReqStatus

verifGetReqs

```
verifGetReqs(
     g_sessionId
     [ ?type t_type ]
     [ ?owner t_owner ]
     [ ?inHier t_parentReqId ]
     [ ?mappable g_mappable ]
     [ ?mapped g_mapped ]
     [ ?domain g_domain ]
     )
     => l_reqs / nil
```

Description

Retrieves the list of requirements from a Verifier session that have the same type, owner, hierarchy, mappable or mapped property, and domain as specified.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
?type *t_type*	Type of requirement.
?owner *t_owner*	Owner of requirement.
?inHier *t_parentReqId*	ID of the parent requirement.
?mappable *g_mappable*	Boolean value that specifies if the requirement must be mappable.
?mapped *g_mapped*	Boolean value that specifies if the requirement must be mapped to the implementation items.
?domain *g_domain*	String value that specifies the domain of the requirements.

Value Returned

l_reqs	A list of requirements, all or specific ones that match the specified criteria.

`nil`	There are no requirements in the Verifier session or no requirements match the conditions.

Examples

The following example opens a Verifier cellview and retrieves the list of the requirements from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Gets all requirements.

```
verifGetReqs(uid)
("ID1" "ID2" "ID3" "ID3.1" "ID3.2")
```

Gets requirements that are of type `Spec Pass`.

```
verifGetReqs(uid ?type "Spec Pass")
=> ("ID3")
```

Gets the unmapped requirements.

```
verifGetReqs(uid ?mapped nil)
=> ("ID2" "ID3" "ID3.1")
```

Gets requirements that are of type `Ran Ok`, belong to `Tom`, are children of parent `ID3`, are mappable and mapped:

```
verifGetReqs(uid ?type "Ran Ok" ?owner "Tom" ?inHier "ID3" ?mappable t ?mapped t
?domain "SIMULATION")
=> ("ID3.2")
```

Related Topics

verifGetReferencedCellViews

verifGetReqParent

verifGetReqProp

verifGetReqProps

verifGetReqStatus

verifGetReqSignoff

```
verifGetReqSignoff(
     g_sessionId
     t_reqId
     )
     => l_reqSignoff / nil
```

Description

Retrieves the signoff setup of the specified requirement in a Verifier session. The retrieved information includes the name of the person who signed off the signoff validity period, comments, and if the signoff is enabled for use or disabled.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.

Value Returned

l_reqSignoff	Disembodied property list (DPL) containing the sign-off setup of the specified requirement.
`nil`	Specified requirement does not have any sign-off setup.

Examples

The following example opens a Verifier cellview and retrieves a requirement that has the signoff setup:

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Retrieves the disembodied property list of signoff data for the specified requirement.

```
dpl = verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 15:25:59 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffComments
"Manully passing this requirement until next run."
)
```

Retrieves the value for `signoffUser`.

```
dpl->signoffUser
```

```
=> "Tom"
```

Related Topics

verifDeleteReqSignoff

verifSignOffReq

verifGetReqStatus

```
verifGetReqStatus(
     g_sessionId
     t_reqId
     )
     => t_status / nil
```

Description

Returns the overall status of a requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_redId	ID of the requirement.

Value Returned

t_status	Overall status of the specified requirement ID. This is the text on *Overall Status* column of the *Results* tree.
`nil`	Overall status of the specified requirement ID is not returned.

Examples

The following example opens a Verifier cellview and retrieves the overall status of the specified requirements.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

```
verifGetReqStatus(uid "ID1")
=> "0%"
verifGetReqStatus(uid "ID1.1")
=> "Not Mapped"
```

Related Topics

verifGetReferencedCellViews

verifGetReqParent

verifGetReqProp

verifGetReqProps

verifGetReqs

verifGetSetupLibrary

```
verifGetSetupLibrary(
     g_sessionId
     )
     => library/cell/view / nil
```

Description

Retrieves the name of the setup library cellview bound to the given Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

"library/cell/view"	A string in the *"library/cell/view"* format which contains the setup library cellview bound to the Verifier session.
`nil`	No setup library cellview is bound to the Verifier session.

Examples

The following example starts a Verifier session which is bound to a setup library cellview and returns the name of the setup library cellview.

Starts a Verifier session

```
uid = verifOpenCellView("Two_Stage_Opamp" "OpAmp" "verifier")
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
```

Returns the setup library cellview name `Two_Stage_Opamp/OpAmp/setupLib` which is bound to the Verifier session `0`.

```
verifGetSetupLibrary(sessionId)
=> "Two_Stage_Opamp/OpAmp/setupLib"
```

Opens a new verifier view without a setup library cellview bound to it.

```
sessionIdNew = verifOpenCellView("Two_Stage_Opamp" "OpAmp" "verifier_new")
INFO (VERIFIER-8215): Started Verifier session '1'.
```

```
=> 1
verifGetSetupLibrary(sessionIdNew)
nil
```

Related Topics

verifSetSetupLibrary

Setup Library Assistant Functions

verifImageExists

```
verifImageExists(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Checks if the given file exists in the `images` directory of the Verifier cellview.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the file.

Value Returned

`t`	The file exists in `images` directory of the Verifier cellview.
`nil`	The file does not exist or the operation is unsuccessful.

Examples

Opens a Verifier cellview and adds an image file from the current directory to the session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.png` from the current directory to the session.

```
verifAddDocument(sess "file.png")
=> "images/file.png"
```

Checks if the file has been added.

```
verifImageExists(sess "file.png")
=> t
```

Returns `nil` if the check is performed for a file that was not added.

```
verifImageExists(sess "another_file.png")
=> nil
```

Related Topics

verifAddDocument

verifAddImage

verifDocumentExists

verifGetDocuments

verifGetImages

verifRemoveDocument

verifRemoveImage

verifImportFile

```
verifImportFile(
     g_sessionId
     t_fileName
     [ ?fileType t_fileType ]
     [ ?headerRows x_headerRows ]
     [ ?columnHeaders l_columnHeaders ]
     [ ?sheetNames l_sheet_Names ]
     [ ?ignoreInavalidRows g_ignoreInvalidRows ]
     )
     =>t / nil
```

Description

Imports the requirements from a CSV or Excel file into a Verifier session. You can import requirements by referencing them or by copying them from the file.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	The name of the Excel or CSV file.
`t_fileType`	The type of the file, that can be "CSV" or "Excel".
`?headerRows` *`x_headerRows`*	The number of the row which contains the column header definitions.
`?columnHeaders` *`l_columnHeaders`*	The list of the header column names that identify the kind of data being loaded from the file and their sequence. The list can contain the following names: `Parent`, `ID`, `Title`, `Type`, `MinSpec`, `MaxSpec`, `Unit`, `Owner`, `Description`. **Note:** Do not use this optional key argument if you specify `?headerRows`.
`?sheetName` *`l_sheetName`*	A list of the names of the Excel sheets to be imported.
`?ignoreInavalidRows` *`g_ignoreInvalidRows`*	Ignores the rows containing invalid properties. The default value is `nil`.

Value Returned

`t`	The requirements are imported from the specified file successfully.
`nil`	The file contains errors or there are no valid requirements in the file.

Examples

The following example opens a Verifier cellview and imports the requirements from a CSV file.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifImportFile(sessionId "test.csv" ?fileType "CSV" ?headerRows 1
?ignoreInvalidRows t)
=> t
```

Related Topics

verifCompareImportedFiles

verifMergeImportedFiles

```
verifMergeImportedFiles(
     g_sessionId
     l_files
     [ ?interactive g_interactive ]
     )
     => t / nil
```

Description

Merges the requirements from the Verifier session with the requirements from the imported files. If differences exist, the *Compare and Merge* form is displayed.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`l_files`	The list of files to be merged. If no files are specified, all imported files are merged.
`?interactive` *`g_interactive`*	*`g_interactive`* accepts one of the following values: ■ `nil`: Requirements are merged (default). ■ `t`: Displays the *Compare and Merge* form if there are differences.

Value Returned

`t`	Files are merged successfully.
`nil`	There are no imported files defined in the specified session.

Examples

The following example opens a Verifier cellview and merges the requirements in the session with the imported requirements from a CSV file:

```
list1 = list("./csv/111.csv")
verifMergeImportedFiles("0" list1)
=> t
```

Related Topics

verifCompareImportedFiles

verifExportJson

verifExportReqsToFile

verifGetImportedFiles

verifImportFile

verifMoveReq

```
verifMoveReq(
     g_sessionId
     g_reqId
     [ ?parentId g_parentId ]
     [ ?pos x_pos ]
     )
     => t / nil
```

Description

Changes the position of a requirement in the specified Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`g_reqId`	ID of the requirement to be moved.
`?parentId g_parentId`	ID of the new parent requirement. Accepts one of the following values: ■ `nil`: Parent does not change (default) ■ `""`: Moves the requirement to the top level ■ ID of a requirement with type `Note` - The specified requirement is moved to be a child of this `Note` requirement.
`?pos x_pos`	The new position of the specified requirement. A value of 0 places the requirement at the beginning of the hierarchy, 1 places it after the first item, 2 places it after the second item, and so on. A negative number, or a number higher than the number of items of the hierarchy, places the requirement at the end of the hierarchy. The default is `-1`.

Value Returned

`t`	The requirement was moved to the required position.
`nil`	The requirement was not moved.

Examples

Open Verifier cellview and move requirement with ID `ID1.1` to be the first child of requirement `ID1`.

```
sessionID = verifOpenCellView("test" "setup" "verifier")
=> 0
verifMoveReq(sessionId "ID1.1" ?parentId "ID1" ?x_pos 0)
=> t
```

Related Topics

verifAddReq

verifRemoveReq

verifRemoveDocument

```
verifRemoveDocument(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Removes the given file from `documents` directory of the Verifier cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	The name of the file to be removed.

Value Returned

`t`	The file is removed successfully.
`nil`	The file does not exist or could not be removed.

Examples

Opens a Verifier cellview and removes a document file from the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.txt` from the current directory to the session.

```
verifAddDocument(sess "file.txt")
=> "documents/file.txt"
verifGetDocuments(sess)
=> ("file.txt")
verifRemoveDocument(sess "file.txt")
=> t
```

Returns `nil` if the file does not exist.

```
verifGetDocuments(sess)
=> nil
verifRemoveDocument(sess "file.txt")
=> nil
```

Related Topics

verifAddDocument

verifAddImage

verifDocumentExists

verifGetDocuments

verifGetImages

verifImageExists

verifRemoveImage

verifRemoveImage

```
verifRemoveImage(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Removes the given image file from the `images` directory of the Verifier cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	The name of the file to be removed.

Value Returned

`t`	The image file is removed successfully.
`nil`	The image file does not exist or could not be removed.

Examples

Opens a Verifier cellview and removes an image file from the current session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds `file.png` from the current directory to the session.

```
verifAddImage(sess "file.png")
=> "images/file.png"
verifGetImages(sess)
=> ("file.png")
verifRemoveImage(sess "file.png") ;
=> t
```

Returns `nil` if the file does not exist.

```
verifGetImages(sess)
=> nil
verifRemoveImage(sess "file.png")
=> nil
```

Related Topics

verifAddDocument

verifAddImage

verifDocumentExists

verifGetDocuments

verifGetImages

verifImageExists

verifRemoveDocument

verifRemoveReq

```
verifRemoveReq(
     g_sessionId
     g_reqId
     )
     => t / nil
```

Description

Deletes one single requirement or a list of requirements in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_reqId	The single requirement ID or list of requirement IDs.

Value Returned

`t`	Specified requirements are deleted from the Verifier session.
`nil`	Specified requirements are not deleted from the Verifier session.

Examples

The following example opens a Verifier cellview and deletes the specified requirements.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Delete a list of requirements:

```
verifRemoveReq(uid list("ID1.1" "ID1.2"))
=> t
```

Deletes a requirement.

```
verifRemoveReq(uid "ID1")
=> t
```

Related Topics

verifAddReq

verifMoveReq

verifSetCustomFieldValue

```
verifSetCustomFieldValue(
     g_sessionId
     t_fieldName
     t_fieldValue
     )
     => l_set / nil
```

Description

Sets the value of a custom field in a Verifier session.

Note: Custom fields can be specified using the Preferences form. To access the Preferences form from the Verifier graphical user interface, choose *Edit — Preferences*.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fieldName`	Name of the custom field.
`t_fieldValue`	Value of the custom field.

Value Returned

`l_set`	Value of the custom field is successfully set to the new value.
`nil`	Value of the custom field is not set.

Examples

Consider that you have defined the custom fields *Manager Name* and *E-mail Address* for a verification project.

The following example opens a Verifier cellview of the project and sets the values in the custom fields.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetCustomFieldNames(uid)
=> ("Email" "VerificationManager")
verifSetCustomFieldValue(uid "VerificationManager" "Harry")
=> t
```

```
verifGetCustomFieldValue(uid "VerificationManager")
=> "Harry"
```

Related Topics

verifGetCustomFieldNames

verifGetCustomFieldValue

verifGetReqCustomFieldNames

verifGetReqCustomFieldValue

verifSetReqCustomFieldValue

verifSetReqCellviewHolder

```
verifSetReqCellviewHolder(
     g_sessionId
     t_reqId
     g_owned
     )
     => t / nil
```

Description

If `g_owned` is specified, sets the 'Cellview Holder' property of specified requirement to current Verifier cellview, else clears the value of the property.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	ID of the requirement.
g_owned	Owned state of the specified requirement.

Value Returned

`t`	Sets 'Cellview Holder' of requirement to current cellview or clears 'Cellview Holder' successfully.
`nil`	vManager is disabled or vManager is not setup in the specified Verifier session.

Examples

The following example starts a Verifier session with a cellview and sets or clears the 'Cellview Holder' property of the specified requirement:

```
uid = verifOpenCellView("test" "example" "verifier")
=>0
verifSetReqCellviewHolder(uid "ID1" t)
=>t
verifGetReqProp(uid "ID1" "cvHolder")
"test example verifier"
verifSetReqCellviewHolder(uid "ID1" nil)
=>t
verifGetReqProp(uid "ID1" "cvHolder")
""
```

Related Topics

verifCreateRandomId

verifSetReqId

verifSetReqTitle

verifSetReqType

verifSetReqProp

verifSetReqCustomFieldValue

```
verifSetReqCustomFieldValue(
     g_sessionId
     t_reqId
     t_fieldName
     t_fieldValue
     )
     => t / nil
```

Description

Sets the custom field value of the specified requirement in a Verifier session.

Note: Custom fields can be specified using the Preferences form. To access the Preferences form from the Verifier graphical user interface, choose *Edit — Preferences*. To access the requirement editor form from the Verifier graphical user interface, select the requirement and choose *Edit — Open Requirement Editor*.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_reqId`	The ID of the requirement.
`t_fieldName`	The name of the custom field.
`t_fieldValue`	The value of the custom field.

Value Returned

`t`	The value of the custom field is successfully set to the new value.
`nil`	The value of the custom field is not set.

Examples

Consider that you have defined the custom fields *Engineer Name* and *E-mail Address* for the requirements of a verification project.

The following example opens a Verifier cellview of the verification project and sets the values in the custom fields for the specified requirement.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetReqCustomFieldNames(uid)
=> ("Email" "Engineer")
verifGetReqCustomFieldValue(uid "ID1" "Engineer")
=> ""
verifSetReqCustomFieldValue(uid "ID1" "Engineer" "Harry")
=> t
verifGetReqCustomFieldValue(uid "ID1" "Engineer")
=> "Harry"
```

Related Topics

verifGetCustomFieldValue

verifGetReqCustomFieldNames

verifGetReqCustomFieldValue

verifSetCustomFieldValue

verifGetCustomFieldNames

verifSetReqId

```
verifSetReqId(
     g_sessionId
     t_reqId
     t_newReqId
     )
     => t / nil
```

Description

Sets the ID property for the specified requirement.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID for the requirement.
t_newReqId	The new ID for the requirement.

Value Returned

`t`	Requirement ID has been updated to *t_newReqId*.
`nil`	Requirement ID has not been updated to *t_newReqId* because of errors.

Examples

The following example opens a Verifier cellview and sets the ID of a requirement to "ID4".

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
verifSetReqId(sessionId "ID3" "ID4")
=> t
```

Related Topics

verifCreateRandomId

verifSetReqCellviewHolder

verifSetReqProp

```
verifSetReqProp(
     g_sessionId
     t_reqId
     t_propertyName
     t_propertyValue
     )
     => t / nil
```

Description

Sets the property for the specified requirement.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID for the requirement.
t_propertyName	Name of the property to be set.
t_propertyValue	New value of the property.

Value Returned

`t`	Property value was set.
`nil`	Property value was not set because of errors.

Examples

The following example opens a Verifier cellview and sets the type to `Note`.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
verifSetReqProp(sessionId "ID3" "type" "Note")
=> t
```

Related Topics

verifCreateRandomId

verifSetReqCellviewHolder

verifSetReqTitle

```
verifSetReqTitle(
     g_sessionId
     t_reqId
     t_newTitle
     )
     => t / nil
```

Description

Sets the ID property for the specified requirement.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID for the requirement.
t_newTitle	New title for the requirement.

Value Returned

`t`	Requirement title has been updated to *t_newTitle*.
`nil`	Requirement title has not been updated to *t_newTitle* because of errors.

Examples

The following example opens a Verifier cellview and sets the title of a requirement to "DC gain".

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
verifSetReqTitle(sessionId "ID3" "DC gain")
=> t
```

Related Topics

verifCreateRandomId

verifSetReqCellviewHolder

verifSetReqType

```
verifSetReqType(
     g_sessionId
     t_reqId
     t_newType
     )
     => t / nil
```

Description

Sets the 'Type' property for the specified requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID for the requirement.
t_newType	The value of the requirement verification type. The type value can be one of the following: ■ `Note` for notes ■ `Spec Pass` for specification checks ■ `Ran Ok` for simulation run status ■ `Manual` for manual sign-off ■ `ExtRef` for external references

Value Returned

`t`	Requirement type has been updated to *t_newType*.
`nil`	Requirement type has not been updated to *t_newType* because of errors.

Examples

The following example opens a Verifier cellview and changes the type of specified requirement.

```
uid = verifOpenCellView("test" "setup" "verifier")
=> 0
verifSetReqType(uid "ID3" "Manual")
=> t
```

Related Topics

verifCreateRandomId

verifSetReqCellviewHolder

verifSetReqId

verifSetReqProp

verifSetReqTitle

verifSetSetupLibrary

```
verifSetSetupLibrary(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Attaches a setup library cellview to a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	String value specifying the library name.
t_cellName	String value specifying the cell name.
t_viewName	String value specifying the view name.

Value Returned

`t`	Attached the given setup library cellview to the given Verifier session.
`nil`	The set operation was unsuccessful. Possible reasons for an unsuccessful operation are: ■ The given setup library cellview does not exist. ■ The given setup library cellview is already attached to the session. ■ The Verifier session is read-only.

Examples

Starts a Verifier session and attaches a setup library to the session.

```
sessionId = verifOpenCellView("Two_Stage_Opamp" "OpAmp" "verifier")
INFO (VERIFIER-8215): Started Verifier session '0'.
```

```
=> 0
```

Returns `t` when the existing setup library cellview `Two_Stage_Opamp/OpAmp/setupLib` is attached to the Verifier session successfully.

```
verifSetSetupLibrary(sessionId "Two_Stage_Opamp" "OpAmp" "setupLib")
=> t
```

Returns `nil` when the setup library cellview `Two_Stage_Opamp/OpAmp/projectSetup` does not exist.

```
verifSetSetupLibrary(sessionId "Two_Stage_Opamp" "OpAmp" "projectSetup")
*WARNING* (VERIFIER-3908): Cannot open SLA cellview Two_Stage_Opamp/OpAmp/
projectSetup because the cellview does not exist.
=> nil
```

Returns `nil` if the Verifier session is read-only:

```
verifSetSetupLibrary(sessionId "Two_Stage_Opamp" "OpAmp_AC_top" "setupLib")
*WARNING* (VERIFIER-1000): cannot change Setup Library to Two_Stage_Opamp/
OpAmp_AC_top/setupLib in read-only session '1'
=> nil
```

Related Topics

verifGetSetupLibrary

Setup Library Assistant Functions

verifSignOffReq

```
verifSignOffReq(
     g_sessionId
     [ ?reqId g_reqId ]
     [ ?userName t_userName ]
     [ ?lifetime t_lifetime ]
     [ ?expDate t_expDate ]
     [ ?comments t_comments ]
     [ ?disableState g_disableState ]
     )
     => t / nil
```

Description

Signs off the specified requirements using the provided information in a Verifier session. You can sign off the requirements that have the verification type 'Manual' or the requirements that failed verification (the overall status of requirement is 'Fail', 'Not Mapped', 'Spec Check Fail', 'Signoff Required' or 'Signed Off').

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?reqId g_reqId`	Single requirement ID or list of requirement IDs. Default is nil and it signs off all requirements. **Note:** The requirement that cannot be signed off will be ignored.
`?userName t_userName`	Name of the user who is signing off the specified requirements.
`?lifetime t_lifetime`	Validity period of the signoff. The supported values are: ■ `Once`: Signoff is valid until next run. ■ `Until Date`: Signoff is valid up to the date specified using the option `?expDate t_expDate`. ■ `Indefinitely`: Signoff is valid for an infinite period.
`?expDate t_expDate`	String value indicating the expiration date of the signoff. Use it to specify the value of `t_lifetime` as '`Until Date`'. Specify the date format as 'MMM d yyyy', such as 'Jan 9 2018' and ensure that the specified date is not earlier than the current date.
`?comments t_comments`	Any comment about the manual signoff.
`?disableState g_disableState`	Status indicating if the signoff is enabled or disabled for use in the verification project.

Value Returned

`t`	Specified requirements are signed off.
`nil`	Specified requirements are not signed off.

Examples

The following example oOpens a Verifier cellview and then signs off the requirements.

```
uid = verifOpenCellView("test" "results" "verifier")
=>0
```

Signs off a single requirement.

```
verifGetReqSignoff(uid "ID1")
=>nil
```

Signs off a single requirement.

```
verifSignOffReq(uid ?reqId "ID1" ?comments "Manually passing this requirement until
next run")
=> t
verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 16:07:56 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffFullUserName
"" signoffExpireDate "" signoffComments "Manually passing this requirement until
next run"
)
```

Signs off a list of requirements.

```
verifSignOffReq(uid ?reqId list("ID1" "ID2"))
=> t
```

Signs off all requirements that can be signed off.

```
verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 16:09:31 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffFullUserName
"" signoffExpireDate "" signoffComments ""
)
```

Related Topics

verifDeleteReqSignoff

verifGetReqSignoff

Custom Fields Functions

Verifier lets you include additional information for your design verification project, and for each requirement in the project. For example, you can include the name and e-mail address of the verification project manager. For each requirement in the project, you can include the name and e-mail address of the engineer as well.

To use this feature, first define the custom field tags, names, and tooltips in a `.csv` file for the project and for the requirements. See the following examples:

The `infoProject.csv` file contains the custom field details for the project:

```
Manager,Manager Name,The name of the project manager.
Email,E-mail Address,The e-mail address of the project manager.
```

The `infoReq.csv` file contains the custom field details for the requirements in the project:

```
Engineer,Engineer Name,The name of the lead engineer.
Email,E-mail Address,The e-mail address of the engineer.
```

Set the Verifier environment variable `customFieldConfig` and `customReqFieldConfig` to indicate the `.csv` files so that Verifier can start using the custom fields. For example, you can set these variables in the `.cdsinit` file, as illustrated below.

```
envSetVal("verifier.customFieldConfig" "csvfile" 'string  "./infoProject.csv")
envSetVal("verifier.customReqFieldConfig" "csvfile" 'string  "./infoReq.csv")
```

Related Topics

verifGetReqCustomFieldNames

verifGetCustomFieldValue

verifGetReqCustomFieldNames

verifGetReqCustomFieldValue

verifSetCustomFieldValue

verifSetReqCustomFieldValue

Requirement Functions

2

Verifier Session and Setup Functions

You can launch a Verifier session from the Virtuoso environment. A Verifier session can have only unique implementation cellview–history entries.

While a Verifier session only supports running one cellview in a Virtuoso session, it is possible to open multiple Verifier sessions in a Virtuoso session. You can open a read-only Verifier cellview in multiple Virtuoso sessions at the same time. However, you can run a simulation in only one of the sessions at a time so that the run summary data of the simulation is not overwritten.

When saving a session, Verifier creates a backup of the `settings.v3` file as `settings.v3~`. When you open a new session, Verifier checks if the current `settings.v3` contains any setup information.

When you initiate the simulation of a local implementation from Verifier, a session is loaded with the implementation cellview where the specified history is set as the active history within that session.

The SKILL functions that let you manage the ADE Verifier setup and session can be broadly classified in the following categories.

- Batch Functions - These functions let you create batch scripts in Verifier.
  - verifCreateBatchScript
  - verifIsBatchRunProcess
- Startup, Exit, and Session Information Functions - These functions let you start and exit Verifier.
  - verifCloseSession
  - verifGetAllSessions
  - verifGetCallbacks
  - verifGetCellViewSession
  - verifGetSessionCellView

- verifGetWindow
- verifIsSessionModified
- verifIsSessionReadOnly
- verifIsValidSession
- verifOpenCellView
- verifRegisterCallback
- verifRemoveCallback
- verifSaveSession
- verifSaveSessionAs

■ Preferences Functions - These functions let you set your Verifier preferences.

- verifGetOptionVal
- verifSetOptionVal

Related Topics

Requirement Functions

Implementation Functions

verifCloseSession

```
verifCloseSession(
     g_sessionId
     [ ?saveIfModified g_saveIfModified ]
     )
     => t / nil
```

Description

Closes a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?saveIfModified` *g_saveIfModified*	A boolean which controls whether to save the session, or discard any changes if it is modified. If any value is not specified and the session has been modified, then a question dialog prompts to either save or discard the changes.

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?saveIfModified` *g_saveIfModified*	A boolean which controls whether to save the session, or discard any changes if it is modified. If any value is not specified and the session has been modified, then a question dialog prompts to either save or discard the changes.

Value Returned

This is a sample lead-in

`t`	Specified Verifier session was closed successfully.
`nil`	Specified Verifier session was not opened or the command was unsuccessful.

Examples

The following example starts a Verifier cellview and adds a requirement. It then saves the cellview and closes the session using `verifCloseSession`.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Closes the unmodified session

```
verifCloseSession(sess)
sess = verifOpenCellView("test" "sample" "verifier")
=> 1
```

Now modifies it by adding a requirement

```
verifAddReq(0 "A Requirement")
=> t
```

Will save then close the session - if `saveIfModified` is set to `nil`, the changes would be discarded before closing.

```
verifCloseSession(sess ?saveIfModified t)
=> t
```

Related Topics

verifSaveSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifCreateBatchScript

```
verifCreateBatchScript(
     g_sessionId
     t_scriptFileName
     )
     => t / nil
```

Description

Creates a batch script to rerun the saved Verifier cellview from the command line.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_scriptFileName	The batch script file name.

Value Returned

`t`	The batch script file is created successfully.
`nil`	The command is not successful.

Examples

The following example opens a Verifier cellview and creates the batch script file for the specified session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifCreateBatchScript(uid "run")
=> t
```

Runs the script.

```
$ virtuoso -replay run -log run.log
```

Related Topics

verifIsBatchRunProcess

Verifier Session and Setup Functions

verifGetAllSessions

```
verifGetAllSessions(
     )
     => l_ids / nil
```

Description

Returns a list of all the open Verifier session IDs.

Arguments

None

Value Returned

l_ids	List of integers, string numbers, or windows specifying the Verifier session IDs.
`nil`	Failed to find any open sessions.

Examples

The following example opens a Verifier cellview and returns all open session IDs.

```
sess = verifOpenCellView("test" "setup" "verifier")
=> 0
verifGetAllSessions()
=> (0)
```

Related Topics

verifCloseSession

verifGetAllSessions

verifGetCellViewSession

verifGetSessionCellView

verifIsValidSession

verifSaveSession

Verifier Session and Setup Functions

verifGetCallbacks

```
verifGetCallbacks(
     )
     => l_callbacks
```

Description

Return the list of callbacks that have been registered by `verifRegisterCallback`(`sess cb`).

Arguments

None

Value Returned

`l_callbacks`	List of registered callbacks.

Examples

The following example starts a Verifier cellview and returns the registered callbacks.

```
defun(myCallback (sess sig args)
info("Callback: %L %L %L\n" sess sig args)
)
=> myCallback

verifRegisterCallback('myCallback)
=> myCallback

verifRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess sig
args)))
=> funobj@0x254db9a8

verifGetCallbacks()
=> (myCallback funobj@0x254db9a8)
```

Related Topics

verifGetCallbacks

verifRegisterCallback

verifRemoveCallback

Verifier Session and Setup Functions

verifGetCellViewSession

```
verifGetCellViewSession(
     t_libName
     t_cellName
     t_viewName
     )
     => x_sessionId / nil
```

Description

Returns the integer session ID for the specified opened Verifier cellview.

Arguments

t_libName	Library name for the opened Verifier cellview.
t_cellName	Cell name for the opened Verifier cellview.
t_viewName	View name for the opened Verifier cellview.

Value Returned

x_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`nil`	Specified Verifier cellview does not open.

Examples

The following example retrieves the session ID of a Verifier cellview.

```
verifOpenCellView("test" "setup" "verifier")
=> 1
sessionId = verifGetCellViewSession("test" "setup" "verifier")
=> 1
```

Related Topics

verifCloseSession

verifGetAllSessions

verifGetSessionCellView

verifIsSessionModified

verifIsValidSession

verifSaveSession

Verifier Session and Setup Functions

verifGetOptions

```
verifGetOptions(
     g_sessionId
     [g_getValues]
     )
     => l_optionNameList / nil
```

Description

Returns the list of Verifier preference options. These options are also available in the Preferences form.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_getValue	When non-nil the return value will be a list of pairs, where each pair will be the name of the option and its current value.

Value Returned

l_optionNameList	The list of all preference options of Verifier.
`nil`	The command is not successful.

Examples

The following example opens a Verifier cellview and obtains the list of preference options.

```
uid=verifOpenCell("myLibrary" "myCell" "verifier")
=> 0
verifGetOptions(uid)
=> ("appendimpinfo" "autocreatereports" "batchhtmlreport" "displaycpk"
"displaymax" "displaymean" "displaymin" "displaysigmatotarget"
"displaystandarddeviation" "displayyield" "enableCoverage" "enablecrossselection"
"impjobpolicy" "impresexpperiod" "impuimode" "linkdatasheets" "mappingonetoone"
"openhtmldialog" "overwritereports" "palsdir" "palswhere" "reqspecs"
"reqspecscheckunits" "simparallelcount" "usereportidenticalhistory"
"weightedAverageForCoverage"
)
```

```
verifGetOptions(uid t)
=>(("appendimpinfo" t)
        ("displaycpk" nil)
        ("displaymax" nil)
        ("displaymean" nil)
        ("displaymin" nil)
        ("displaysigmatotarget" nil)
        ("displaystandarddeviation" nil)
        ("displayyield" nil)
        ("enableCoverage" t)
        ("enablecrossselection" t)
        ("impjobpolicy" "")
        ("impresexpperiod" 0)
        ("impuimode" nil)
        ("linkdatasheets" nil)
        ("mappingonetoone" nil)
        ("openhtmldialog" nil)
        ("overwritereports" t)
        ("palsdir" "")
        ("palswhere" "currentcellview")
        ("reqspecs" "Use Requirement Spec and Check")
        ("reqspecscheckunits" t)
        ("simparallelcount" 1)
        ("usereportidenticalhistory" nil)
        ("weightedAverageForCoverage" t)
    )
```

Related Topics

verifGetOptionVal

verifSetOptionVal

Verifier Session and Setup Functions

verifGetOptionVal

```
verifGetOptionVal(
     g_sessionId
     t_optionName
     )
     => t_optionValue / nil
```

Description

Returns the value of a preference option set in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_optionName	The name of the preference option. The name must be one of the values returned by verifSetOptionVal.

Value Returned

l_optionValue	The value of the specified preference option.
`nil`	The specified option does not have a value, or the command is not successful.

Examples

The following example opens a Verifier cellview, and retrieves the values of the specified options. For options that are switched on or off, Verifier returns `t` or `nil`, respectively. For options with other values, Verifier returns the set values.

```
uid=verifOpenCell("test" "sample" "verifier")
=> 0
verifGetOptionVal(uid "enableCoverage")
=> t
verifGetOptionVal(uid "simparallelcount")
=> 2
verifGetOptionVal(uid "reqspecs")
=> "Use Requirement Spec and Check"
```

Related Topics

verifGetOptions

verifSetOptionVal

Verifier Session and Setup Functions

verifGetSessionCellView

```
verifGetSessionCellView(
     g_sessionId
     )
     => l_libCellView / nil
```

Description

Returns the cellview name for the specified session.

Argument

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`l_libCellView`	A list containing the library, cell and view for the specified session.
`nil`	The session does not exist.

Examples

The following example retrieves the library, cell, and view of a newly opened Verifier session.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
verifGetSessionCellView(sessionId)
("test" "setup" "verifier")
```

Related Topics

verifGetAllSessions

verifGetSessionCellView

verifIsSessionModified

verifIsSessionReadOnly

verifIsValidSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifGetWindow

```
verifGetWindow(
     g_sessionId
     )
     => w_windowId / nil
```

Description

Returns the window ID related to the specified session ID.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

w_windowId	Window ID of the specified session.
`nil`	Session does not exist or there is no corresponding window.

Examples

The following example retrieves the window ID of a newly created Verifier cellview.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
windowID = verifGetWindow(sessionId)
=> window:2
```

Related Topics

verifGetWindow

verifIsSessionModified

verifIsSessionReadOnly

verifIsValidSession

verifOpenCellView

verifSaveSession

Verifier Session and Setup Functions

verifIsBatchRunProcess

```
verifIsBatchRunProcess(
     )
     => t / nil
```

Description

Returns `t` if the code is currently running in a remote child process for ADE Verifier. You can use this function in your `.cdsinit` file or in custom SKILL code.

Arguments

None

Value Returned

`t`	Child IC remote process is running.
`nil`	Child IC remote process is not running.

Examples

The following example shows how to use this function.

In Virtuoso CIW:

```
axlIsICRPProcess( )
=> nil
```

In the `.cdsinit` file:

```
if(verifIsBatchRunProcess() then
    info("## Is batch run process ##\n")
    ;; Do something for batch run here
else
    info("## Is not batch run process ##\n")
    ;; Do something for non-batch run here
)
```

Related Topics

verifCreateBatchScript

Verifier Session and Setup Functions

verifIsSessionModified

```
verifIsSessionModified(
     g_sessionId
     )
     => t / nil
```

Description

Checks whether the setup has been modified after the last time it was saved in the given session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	The setup of the specified session has been modified after the last time it was saved.
`nil`	The specified session does not exist, or the setup of the specified session has not been modified after the last time it was saved.

Examples

The following example shows how to use this function.

```
uid = verifOpenCellView("test" "new" "verifier")
=> 0

verifIsSessionModified(uid)
=> nil
```

Adds a new requirement

```
verifAddReq(uid "MyReq")
=> "3300663f-d165-4796-8518-b103cb7490d2"

verifIsSessionModified(uid)
=> t
```

Related Topics

verifCloseSession

verifGetCellViewSession

verifGetSessionCellView

verifIsSessionReadOnly

verifIsValidSession

verifSaveSession

Verifier Session and Setup Functions

verifIsSessionReadOnly

```
verifIsSessionReadOnly(
     g_sessionId
     )
     => t / nil
```

Description

Determines if the given session is read-only.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	The session is open read-only.
`nil`	The session is open for editing or does not exist.

Examples

The following example shows how to use this function.

```
sess = verifOpenCellView("test" "sample" "verifier" ?mode "r")
=> 1
verifIsSessionReadOnly(sess)
=> t
verifIsSessionReadOnly(99)
*WARNING* (VERIFIER-5004): verifIsSessionReadOnly - no such session: 99
=> nil
```

Related Topics

verifGetAllSessions

verifIsSessionModified

verifIsValidSession

verifSaveSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifIsValidSession

```
verifIsValidSession(
     g_sessionId
     )
     => t / nil
```

Description

Confirms if the given session is a valid ADE Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	The specified session is a valid ADE Verifier session.
`nil`	The specified session is not a valid ADE Verifier session.

Examples

The following example shows how to use this function.

```
uid = verifOpenCellView("test" "new" "verifier")
=> 0
verifIsValidSession(uid)
=> t
verifIsValidSession("0")
=> t
verifIsValidSession(hiGetCurrentWindow())
=> t
verifIsValidSession(1)
=> nil
```

Related Topics

verifCloseSession

verifIsSessionModified

verifIsSessionReadOnly

verifSaveSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifOpenCellView

```
verifOpenCellView(
     t_libName
     t_cellName
     t_viewName
     [ ?mode g_mode ]
     [ ?openWindow g_openWindow ]
     )
     => x_sessionId / nil
```

Description

Opens the Verifier cellview and returns the session ID.

Arguments

t_libName	The library name of the cellview.
t_cellName	The cell name.
t_viewName	The view name.
`?mode` *g_mode*	The mode in which the cellview is opened. The available options are: `nil`: The session will be opened either for editing or as read-only depending on whether the cellview is editable or not. `r`: The session will be opened as read-only. `w`: The session will be opened with write-access. This will create a new cellview if it does not exist, or overwrite an existing cellview. `a`: The session will be opened for edit.
`?openWindow` *g_openWindow*	Boolean to specify whether to open a window in the cellview or not. If it is `nil` then no window is opened, but the session is still opened in non-graphical mode. The default value is `t`.

Value Returned

x_sessionId	The session ID of the opened cellview.

`nil`	The specified cellview does not open successfully.

Examples

The following example opens a Verifier cellview in a window.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

If a window is already open, the function brings the window to the front and returns the session ID.

```
sess2 = verifOpenCellView("test" "sample" "verifier")
=> 0
```

In the Design Management flow, if a window is already open, the function returns `nil`. No window is opened, but the session is still opened in non-graphical mode.

```
sess2 = verifOpenCellView("test" "sample" "verifier")
=> nil
```

Creates a new cellview.

```
sess3 = verifOpenCellView("test" "sample" "verifier2" ?mode "w")
=> 1
verifSaveSession(sess3)
verifCloseSession(sess3)
```

Reopens the cellview for edit without a window.

```
sess3 = verifOpenCellView("test" "sample" "verifier2" ?mode "a" ?openWindow nil)
=> 2
```

Related Topics

verifCloseSession

verifSaveSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifRegisterCallback

```
verifRegisterCallback(
     g_callback
     )
     => g_callback
```

Description

Registers a callback with Verifier, which is executed at various times, such as, when a session is opened, saved, closed, simulations are started, finished, and when percentage changes.

Callback registration can be done at any time, and even before any Verifier sessions are opened.

The callback function takes three arguments, *`x_session`*, *`x_signal`*, and *`l_args`*. The *`x_session`* is the session identifier that calls the callback. *`x_signal`* is a symbol containing the type of callback, and `args` is a list containing data specific to the signal. For example,

for the `'sessionSaved` signal the `args` is a list:

`'("opamp090.full_diff_opamp_AC:verifier")`

whereas for `'simulationPercentage` the list is:

`'("opamp090/full_diff_opamp_AC/maestro/Active" 49 98 200), and for 'requirementStatusChange it is '("ID6" "Passed").`

The following table describes the supported signals:

Signal	**args**	**When the signal is sent**
`'sessionCreated`		
	`session_num`	When a session is first created either when it is opened from a cellview, or created from scratch. Example: `'(0)`
`'sessionDeleted`		
	`session_num`	When the session has been deleted, which is typically done by closing the window. Example: `'(0)`

`Signal`	args	When the signal is sent
`'sessionSaved`		
	`lib_Name` *`cell_Name`* *`view_Name`*	When the session is saved to disk. Example: `'("myLib" "myCell" "verifier")`
`'sessionLoaded`		
	`lib_Name` *`cell_Name`* *`view_Name`*	When the session has been loaded from disk. Example: `'("myLib" "myCell" "verifier")`
`'simulationStatus`		
	`imp_name` *`sim_status`*	When the simulation status of the given implementation changes. The *`sim_status`* is a string such as "`Running`", "`Finished`". Example: `'("myLib/myCell/maestro/Active" "Finished")`
`'simulationPercentage`		
	`imp_name` *`int_percentage_Complete`* *`int_num_Completed`* *`int_num_Submitted`*	When the implementation simulation percentage changes. The three numbers are the calculated percentage complete, the number of points that have completed, and the total number of points. Example: `'("myLib/myCell/maestro/Active" 75 3 4).`
`'simulationsDone`		
		When all of the simulations for the implementations have completed. Example: `nil`.

`Signal` **args**	**When the signal is sent**
`'simulationsAdded`	
`Int_number_added`	The number of implementations that have been added to the simulation queue. Typically this will happen when the user clicks *Run*. Example: `'(3).`
`'implementationAdded`	
`imp_name`	The name of the implementation that has been added. Typically sent out when an implementation is added to the Verifier session. Example: `'("opamp090/full_diff_opamp_AC/maestro/MonteCarlo.1").`
`'implementationDeleted`	
`imp_name`	The name of the implementation that is being deleted. Typically gets called just before the implementation is deleted so that the implementation to be deleted is still accessible. Example: `'("opamp090/full_diff_opamp_AC/maestro/MonteCarlo.1").`
`'preImplementationModified`	
`imp_name` *`name_property`* *`key_property_value`*	The given implementation is about to be changed. Example: `'("opamp090/full_diff_opamp_AC_bad/maestro/Interactive.226" "Run" t).`
`'implementationModified`	
`imp_name` *`name_property`*	The callback after the implementation has changed. Example: `'("opamp090/full_diff_opamp_AC_bad/maestro/Interactive.226" "Run" nil).`

`Signal`	**args**	**When the signal is sent**
`'mappingCreated`		
	`req_id` *`mappable_name`*	The callback when a mapping between a requirement and implementation is created. Example: `'("z_48" "avgOutN::Cpk").`
`'mappingDeleted`		
	`req_id` *`mappable_name`*	The callback when a mapping between a requirement and implementation is deleted. Typically gets called just before the mapping is deleted so that the mapping to be deleted is still accessible. Example: `'("z_48" "avgOutN::Cpk").`
`'requirementCreated`		
	`req_id`	The ID of the requirement that has been created. Example: `'("2902d5a4-7ec5-4302-b083-3750dfc10da7").`
`'requirementDeleted`		
	`req_id`	The ID of the requirement that is about to be deleted. Typically gets called just before the requirement is deleted so that the requirement to be deleted is still accessible. Example: `'("2902d5a4-7ec5-4302-b083-3750dfc10da7").`

Signal	args	When the signal is sent
`'requirementModified`		
	`req_id` *`property_name`* *`property_value`*	This callback gets a list of changes that match the property name or value specified using `verifGetReqProps()`. Example: `'(("x_5" "MinSpec" "0.5"))`. The only exception to this behavior is when the ID changes and both the IDs, before and after the change, are returned. Example: `'("my_req" "ID" "2902d5a4-7ec5-4302-b083-3750dfc10da7" "my_req")`.

Arguments

`g_callback`	A symbol or function object, where the symbol is the name of a function that takes three arguments, and the function object is a callable function that also takes three arguments.

Value Returned

`g_callback`	The callback is registered.

Examples

The following example shows how to register a callback with Verifier.

```
defun(myCallback (sess sig args)
info("Callback: %L %L %L\n" sess sig args)
)
myCallback

verifRegisterCallback('myCallback)
=> myCallback

verifRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess sig
args)))
funobj@0x254db9a8
```

Related Topics

verifGetCallbacks

verifRemoveCallback

Verifier Session and Setup Functions

verifRemoveCallback

```
verifRemoveCallback(
     g_callback
     )
     => t / nil
```

Description

Remove a callback that was previously registered using `verifRegisterCallback(cb)`.

Arguments

`g_callback`	A symbol or function object.

Value Returned

`t`	The callback is removed.
`nil`	The callback is not registered and therefore is not removed.

Examples

The following example shows how to remove a registered callback from Verifier.

```
defun(myCallback (sess sig args)

info("Callback: %L %L %L\n" sess sig args)
)
myCallback

verifRegisterCallback('myCallback)
myCallback

fn = verifRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess
sig args)))
funobj@0x254db9a8

verifRemoveCallback('myCallback)
=> t
verifRemoveCallback('myCallback)
=> nil
verifRemoveCallback(fn)
=> t
```

Related Topics

verifGetCallbacks

verifRegisterCallback

Verifier Session and Setup Functions

verifSaveSession

```
verifSaveSession(
     g_sessionId
     )
     => t / nil
```

Description

Saves the active setup of the specified Verifier session to the cellview from where the setup was loaded.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Successfully saved the session.
`nil`	Failed to save the session.

Examples

The following example shows how to save a Verifier session in its cellview.

```
sess = verifOpenCellView("test" "setup" "verifier")
=> 0
verifSaveSession(sess)
INFO (VERIFIER-1507): Saved cellview test.setup:verifier
=> t
```

Related Topics

verifCloseSession

verifGetSessionCellView

verifIsValidSession

verifSaveSessionAs

Verifier Session and Setup Functions

verifSaveSessionAs

```
verifSaveSessionAs(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Saves the active setup of specified Verifier session as a cellview of the type ‘verifier’.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	The library name. This must already exist.
t_cellName	The new cell name.
t_viewName	The new view name.

Value Returned

`t`	Session was successfully saved as a new cellview.
`nil`	Failed to save the cellview.

Examples

The following example shows how to save a Verifier session to a new cellview.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 1
verifSaveSessionAs(sess "test" "sample" "verifier_new")
=> t
```

Related Topics

verifCloseSession

verifIsSessionModified

verifSetOptionVal

```
verifSetOptionVal(
     g_sessionId
     t_optionName
     g_value
     )
     => t / nil
```

Description

Sets the value of the specified preference option in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_optionName	The name of the option.
g_value	The value to assign to the option.

Value Returned

t	Successfully sets the new value.
`nil`	The specified option does not exist or the value is invalid.

Examples

The following example shows how to set the value of a preference option.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifSetOptionVal(sess "simparallelcount" 4)
=> t
verifSetOptionVal(0 "displaymax" t)
=> t
verifSetOptionVal(sess "simparallelcount" "abc")
*WARNING* (VERIFIER-1803): verifSetOption: invalid value '"abc"' for integer option
- check the value and try again.
=> nil
verifSetOptionVal(sess "xxx" "1")
*WARNING* (VERIFIER-1800): verifSetOption: unknown option: 'xxx'
=> nil
```

Related Topics

verifGetOptions

verifGetOptionVal

Verifier Session and Setup Functions

3

Implementation Functions

Implementations are `maestro` cellviews, and their tests and outputs that you map to their corresponding requirements. Verifier lets you add implementation cellviews, which you can rearrange, and through which you can search and filter. You can also open the implementation cellviews in their application, such as ADE Assembler.

You can run implementation cellviews from Verifier to capture results, or load existing run results. Verifier uses these results, along with other settings, to determine the verification status.

The SKILL functions that let you manage implementations in the ADE Verifier environment can be broadly classified in the following categories.

- Implementation Management Functions - These functions let you add, move, and delete implementations.
  - verifAddCustomOutput
  - verifAddImp
  - verifGetImpEstRunTime
  - verifGetImpPriority
  - verifSetImpSetPreRunScript
  - verifGetMappableType
  - verifMoveImp
  - verifOverwriteSpec
  - verifRemoveImp
  - verifRestart
  - verifSetImpData
  - verifSetImpEstRunTime

  - verifSetImpPriority
  - verifSetImpSetPreRunScript
  - verifUpdateImpEstRunTime

- Data Extraction Functions - These functions let you extract data from implementations.
  - verifExportCoverageDataForImp
  - verifExportCoverageDataForReq
  - verifGetImpData
  - verifGetImps
  - verifGetImpTestOutputs
  - verifGetImpTests
  - verifGetTypicalSetup
  - verifUpdate

Related Topics

Requirement Functions

Requirement-to-Implementation Mapping Functions

Verifier Session and Setup Functions

verifAddCustomOutput

```
verifAddCustomOutput(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     t_historyName
     t_testName
     t_outputName
     [ ?createRequirements g_create ]
     [ ?data g_data ]
     )
     => t / nil
```

Description

Adds an output to a custom implementation.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	Library name of the existing custom implementation.
t_cellName	Cell name of the existing custom implementation.
t_viewName	View name of the existing custom implementation.
t_historyName	History name of the existing custom implementation.
t_testName	Test name in the existing custom implementation.
t_outputName	Output name in the existing custom implementation.
`[ ?createRequirements` *g_create* `]`	
	Boolean that controls whether requirements must be created and mapped to the implementation outputs.
`[ ?data` *g_data* `]`	List of key or value pairs that must be set on the resulting output. For more details, see `verifGetImpTestOutputs` and `verifSetImpData`.

Value Returned

`t`	The output was successfully added.
`nil`	The output was not added because of an error.

Examples

The following example shows how to add a custom output to an implementation in various scenarios.

Adds custom outputs to an implementation.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifAddCustomOutput(sess "customLib" "cell" "view" "hist" "test1" "out1")
verifAddCustomOutput(sess "customLib" "cell" "view" "hist" "test2" "out2"
?createRequirements t ?data '("details" "45" "unit" "V" "spec" "> 40")
```

Creates a custom implementation view.

```
verifAddImp(sess "l" "c" "v" "h" ?custom t)
```

Adds a test and output to the custom implementation.

```
verifAddCustomOutput(sess "l" "c" "v" "h" "test" "output" ?createRequirements nil)
```

Adds a test and output to the custom implementation with additional data.

```
verifAddCustomOutput(sess "l" "c" "v" "h" "test" "output" ?createRequirements nil
?data list("details" "45" "unit" "V" "spec" ">45")
```

Related Topics

verifAddImp

verifGetImpTestOutputs

verifRemoveImp

verifSetImpData

Implementation Functions

verifAddImp

```
verifAddImp(
     g_sessionId
     t_lib
     t_cell
     t_view
     t_history
     [ ?runState g_runState ]
     [ ?runPlanState g_runPlanState ]
     [ ?createRequirements g_createRequirements ]
     [ ?custom g_custom ]
     )
     => t / nil
```

Description

Adds a new implementation, which is a `maestro` cellview, to the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_lib	The library name of the `maestro` cellview added to the specified Verifier session.
t_cell	The cell name of the `maestro` cellview added to the specified Verifier session.
t_view	The view name of the `maestro` cellview added to the specified Verifier session.
t_history	The history name of the `maestro` cellview added to the specified Verifier session. If you do not specify the history name, Verifier uses the `Active` history of the `maestro` cellview.

`?runState` *`g_runState`*

Sets the check state of the *Run* check box while adding the new implementation to the specified Verifier session. The default value is `t`.

`?runPlanState` *`g_runPlanState`*

Sets the check state of the *Run Plan* check box while adding the new implementation to the specified Verifier session. The default value is `nil`.

`?createRequirements` *`g_createRequirements`*

If enabled, creates the requirements for the implementation hierarchically. The default value is `nil`.

`?custom` *`g_custom`*

If enabled, adds a test or output to a custom implementation. The default value is `nil`.

Value Returned

`t`	The `maestro` cellview is added to the specified Verifier session.
`nil`	The `maestro` cellview is not added to the specified Verifier session.

Examples

The following example opens a Verifier cellview and adds an implementation cellview.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds the given implementation to the session.

```
verifAddImp("0" "opamp090" "full_diff_opamp_AC" "maestro" "Active")
=> t
```

Adds a second implementation to the session, but does not enable the *Run* check box and create requirements from it.

```
verifAddImp(0 "opamp090" "full_diff_opamp_AC" "maestro" "Interactive.1" ?runState
nil ?createRequirements t)
=> t
```

Enables the *Run* check box.

```
verifAddImp(uid "opamp090" "full_diff_opamp" "maestro" "Active" ?runState t)
Verifier: Adding implementation cellview opamp090/full_diff_opamp/maestro
=> t
```

Disables the *Run* check box.

```
verifAddImp(uid "opamp090" "full_diff_opamp" "maestro" "Active" ?runState nil)
Verifier: Adding implementation cellview opamp090/full_diff_opamp/maestro
```

```
=> t
```

Adds a test or output to a custom implementation.

```
verifAddImp(sess "CustomL" "C" "V" "H" ?custom t)
```

Related Topics

verifAddCustomOutput

verifMoveImp

verifOverwriteSpec

verifRemoveImp

verifSetImpData

Implementation Functions

verifExportCoverageDataForImp

```
verifExportCoverageDataForImp(
     g_sessionId
     t_fileName
     t_spaceName
     [ ?implementation g_implementation ]
     [ ?indented g_indented ]
     )
     => t / nil
```

Description

Exports coverage data for the specified implementation.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the JSON file to which the implementation coverage data is exported.
t_spaceName	The name of verification space which is defined in the Setup Library assistant
`?implementation` *g_implementation*	The name of an implementation which can be either a string, such as `Lib/Cell/View/History` or a list `'("Lib" "Cell" "View" "History")`. If no implementation is specified, all implementations in the session are exported.
`?indented` *g_indented*	If enabled, exports the data to the JSON in indented format, which is printed in multiple lines. Otherwise, the data is exported to the JSON in a compact format. The default value is `nil`.

Value Returned

`t`	Coverage data of the specified implementations is exported to the JSON file.
`nil`	Coverage data is not exported.

Examples

The following example exports coverage data for the implementations in the specified session.

Opens an existing Verifier cellview that includes a Setup Library view.

```
uid = verifOpenCellView("test" "coverages" "verifier" ?openWindow nil)
=> 0
```

Export all implementations in compact format

```
verifExportCoverageDataForImp(uid "allImps.json" "FullSpace")
```

```
=> t
```

Exports the specified implementations in indented format

```
verifExportCoverageDataForImp(uid "imp.json" "FullSpace" ?implementation "test/
coverages/maestro_space/Active" ?indented t)
```

```
=> t
```

Exports the specified implementations in compact format

```
verifExportCoverageDataForImp(uid "imp.json" "FullSpace" ?implementation
list("test" "coverages" "maestro_space" "Active"))
```

```
=> t
```

Related Topics

verifExportCoverageDataForReq

verifAddCustomOutput

verifMoveImp

verifOverwriteSpec

verifRemoveImp

verifSetImpData

Implementation Functions

verifExportCoverageDataForReq

```
verifExportCoverageDataForReq(
     g_sessionId
     t_fileName
     [ ?reqId t_reqId ]
     [ ?indented g_indented ]
     )
     => t / nil
```

Description

Exports coverage data for the specified requirement.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the JSON file to which the requirement coverage data is exported.
`?reqId` *t_reqId*	The requirement ID.
`?indented` *g_indented*	If enabled, exports the data to the JSON in indented format, which is printed in multiple lines. Otherwise, the data is exported to the JSON in a compact format. The default value is `nil`.

Value Returned

`t`	Coverage data of the specified requirements is exported to the JSON file.
`nil`	Coverage data is not exported.

Examples

The following example exports coverage data for the requirements in the specified session.

Opens an existing Verifier cellview that includes a requirement with verification space.

```
uid = verifOpenCellView("test" "coverages" "verifier" ?openWindow nil)
=> 0
```

Export all requirements in compact format

```
verifExportCoverageDataForReq(uid "allReqs.json")
```

```
=> t
```

Exports the specified requirements in indented format

```
verifExportCoverageDataForReq(uid "req.json" ?reqId "3a7333a6-d5bb-49a8-9d62-
0048bc92e5cb" ?indented t)
```

```
=> t
```

Related Topics

verifExportCoverageDataForImp

verifAddCustomOutput

verifMoveImp

verifOverwriteSpec

verifRemoveImp

verifSetImpData

Implementation Functions

verifGetImpData

```
verifGetImpData(
     g_sessionId
     g_impLib
     g_impCell
     g_impView
     g_impHistory
     [ ?runName t_runName ]
     [ ?testName t_testName ]
     [ ?outputName t_outputName ]
     [ ?statName t_statName ]
     [ ?dataName t_dataName ]
     )
     => o_dataTable/ t_dataValue/ nil
```

Description

Get the data for an implementation. The common data is `type`, `name` and `map`. For an implementation, the additional data is `runState`, `runPlanState`, and `runMode`. In case of runs, the additional data is `runMode`. Similarly, in case of an output, the additional data is `spec` and `unit` and in case of statistical output, it is `statType`.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2).`
t_impLib	The library name of the implementation.
t_impCell	The cell name of the implementation.
t_impView	The view name of the implementation.
t_impHistory	The history name of the implementation.

`?runName` *`t_runName`*

The name of the implementation run plan run.

`?testName` *`t_testName`*

The name of the implementation test.

`?outputName` *`t_outputName`*

The name of the implementation output.

`?statName` *`t_statName`*

The name of the implementation statistical output.

`?dataName` *`t_dataName`*

The data name is the key in each table element.

Value Returned

`o_dataTable`	Table of implementation object data.
`t_dataValue`	String value for the specified data name.
`nil`	The implementation does not exist or the operation was unsuccessful.

Examples

The following example opens a Verifier cellview and retrieves the implementation object data.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Gets data for an implementation.

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"))
=> (("runMode" "Monte Carlo Sampling")
    ("name" "Two_Stage_Opamp/OpAmp/maestro_MC/Active")
    ("runState" "false")
    ("type" "Implementation")
    ("map" "8509cd48-5cb7-4f65-a975-048afb5a32ef")
    )
```

Gets data for an implementation test.

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC"))
=> (("name" "Two_Stage_Opamp/OpAmp/maestro_MC/Active/AC")
    ("type" "Test")
    ("map" "863951f4-3424-4315-b6fc-edeadbed2b1f")
    )
```

Gets data for an implementation output.

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC" ?outputName "Current"))
=> (("unit" "A")
    ("name" "Two_Stage_Opamp/OpAmp/maestro_MC/Active/AC/Current")
    ("spec" "< 1.5m")
    ("type" "Output")
    ("map" "dac5ef00-cbfa-4e7a-b19d-6a089e776efc")
    )
```

Gets data for an implementation statistical output.

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC" ?outputName "Current" ?statName "Current::Mean"))
```

```
=> (("name" "Two_Stage_Opamp/OpAmp/maestro_MC/Active/AC/Current/Current::Mean")
     ("statType" "Mean")
     ("type" "StatOutput")
     ("map" "87d04770-c703-456c-be78-99afd4dbb447")
    )
```

Gets data for an implementation run.

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_run_plan"
"Active" ?runName "Nominal"))
=> (("runMode" "Single Run, Sweeps and Corners")
     ("name" "Two_Stage_Opamp/OpAmp/maestro_run_plan/Active/Nominal")
     ("type" "Run")
    )
```

Gets data value based on data name.

```
verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active"
?dataName "runMode")
=> "Run Plan"
```

Related Topics

verifGetImpEstRunTime

verifGetImpPriority

verifGetImps

verifSetImpSetPreRunScript

verifGetMappableType

verifGetTypicalSetup

Implementation Functions

verifGetImpEstRunTime

```
verifGetImpEstRunTime(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     )
     => x_estRunTime / nil
```

Description

Retrieves the `EstRunTime` of specified implementation in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	Library name of the implementation.
t_impCell	Cell name of the implementation.
t_impView	View name of the implementation.
t_impHistory	History name of the implementation.

Value Returned

x_estRunTime	Integer number of seconds for estimated run time.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and retrieves the `EstRunTime` of specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0
verifGetImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=>55
```

Related Topics

verifSetImpEstRunTime

verifUpdateImpEstRunTime

Implementation Functions

verifGetImpPriority

```
verifGetImpPriority(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     )
     => x_priority / nil
```

Description

Gets the 'Priority' of the specified implementation in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	Library name of the implementation.
t_impCell	Cell name of the implementation.
t_impView	View name of the implementation.
t_impHistory	History name of the implementation.

Value Returned

x_priority	Integer number for priority.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and retrieves the `Priority` of specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0
verifGetImpPriority(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=>70
```

Related Topics

verifGetImps

verifGetImpEstRunTime

verifSetImpSetPreRunScript

verifGetMappableType

verifGetImpTests

verifGetTypicalSetup

Implementation Functions

verifGetImps

```
verifGetImps(
     g_sessionId
     [ ?library t_impLib ]
     [ ?cell t_impCell ]
     [ ?view t_impView ]
     [ ?history t_impHistory ]
     [ ?runState g_runState ]
     [ ?runPlanState g_runPlanState ]
     [ ?extRef g_extRef ]
     [ ?priority x_priority ]
     [ ?estRunTime x_estRunTime ]
     [ ?custom g_custom ]
     )
     => l_imps / nil
```

Description

Retrieves the list of the implementations from a Verifier session that have the same library, cell, view, or history name as specified, the same check state for the 'Run' or 'RunPlan' check box.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?library` *t_impLib*	The library name of the implementation.
`?cell` *t_impCell*	The cell name of the implementation.
`?view` *t_impView*	The view name of the implementation.
`?history` *t_impHistory*	The history name of the implementation.
`?runState` *g_runState*	The check state of the 'Run' check box.
`?runPlanState` *g_runPlanState*	The check state of the 'RunPlan' check box.
`?extRef` *g_extRef*	The referenced state of the implementation.
`?priority` *x_priority*	The integer number of implementation priority.

`?estRunTime` *`x_estRunTime`*	The integer number of implementation estimated run time.
`?custom` *`g_custom`*	The filtered list of custom implementations.

Value Returned

`l_imps`	The list of implementations that match the conditions.
`nil`	There are no matched implementations or the command is not successful.

Examples

The following example opens a Verifier cellview and retrieves the implementations.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Gets all implementations.

```
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active")
("amsPLL" "CP_TB" "maestro" "Active")
)
```

Gets the implementations belonging to library `amsPLL`.

```
verifGetImps(uid ?library "amsPLL")
(("amsPLL" "CP_TB" "maestro" "Active"))
```

Gets the implementation with enabled 'Run' check box.

```
verifGetImps(uid ?runState t)
(("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
("amsPLL" "CP_TB" "maestro" "Active")
)
```

Gets the implementations that belong to cell `OpAmp` and have the *RunPlan* check box disabled.

```
verifGetImps(uid ?cell "OpAmp" ?runPlanState nil)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
)
```

Gets the referenced implementations.

```
verifGetImps(uid ?extRef t)
=> nil
```

Gets the implementations that have priority equal to `50`:

```
verifGetImps(uid ?priority 50)
(("Two_Stage_Opamp" "OpAmp" "maestro_sweeps-corners" "Active"))
```

Get the implementations that have priority greater than `40`.

```
verifGetImps(uid ?priority lambda((x) x>40))
(("Two_Stage_Opamp" "OpAmp" "maestro_sweeps-corners" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
)
```

Related Topics

verifGetImpData

verifGetImpTestOutputs

verifGetImpTests

Implementation Functions

verifGetImpSetPreRunScript

```
verifGetImpSetPreRunScript(
     g_sessionId
     t_impSetName
     )
     => t_preRunScriptFileName
```

Description

Retrieves the pre-run script file name specified for an implementation set.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	Name of the implementation set.

Value Returned

t_preRunScriptFileName	File name for pre-run script attached to the given implementation set.
`nil`	Implementation does not exist in the Verifier session, or no script is set for the given implementation set.

Examples

The following example starts a Verifier session with a cellview and retrieves the pre-run script file name specified for an implementation set.

```
uid = verifOpenCellView("test" "pre-run" "verifier")
=>0
verifGetImpSetPreRunScript(uid "Weekly")
=>"prerun.il"
```

Related Topics

verifSetImpSetPreRunScript

Implementation Functions

verifGetImpTestOutputs

```
verifGetImpTestOutputs(
     g_sessionId
     t_lib
     t_cell
     t_view
     t_history
     t_test
     [ t_runName ]
     )
     => l_impTestOutputs / nil
```

Description

Returns the list of the outputs for an implementation test in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_lib	Library name of the implementation.
t_cell	Cell name of the implementation.
t_view	View name of the implementation.
t_history	History name of the implementation.
t_test	Name of the test in the specified implementation.
t_runName	Run name for the implementation is in 'Run Plan' mode. This argument is optional.

Value Returned

l_impTestOutputs	The list of the outputs in the specified test.
`nil`	The specified test does not exist or there is no outputs in the specified test.

Examples

The following example opens a Verifier cellview and retrieves all the outputs of a test of the specified implementation.

```
uid = verifOpenCell("test" "sample" "verifier")
=>t
tests=verifGetImpTests(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")
("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
outputs=verifGetImpTestOutputs(uid "opamp090" "full_diff_opamp_AC" "maestro"
"Active" "opamp090:full_diff_opamp_AC:1")
("Bandwidth" "Current" "DCGain" "InputRandomOffset" "MAC_DCGain")
```

For an example on how `verifGetImpTestOutputs` can be used in a custom function, see Copy Implementation Units to Requirements.

Related Topics

verifGetImpTests

Implementation Functions

verifGetImpTests

```
verifGetImpTests(
     g_sessionId
     t_lib
     t_cell
     t_view
     t_history
     [ t_runName ]
     )
     => l_impTests / nil
```

Description

Returns the list of all the tests for an implementation in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_lib	The library name of the implementation.
t_cell	The cell name of the implementation.
t_view	The view name of the implementation.
t_history	The history name of the implementation.
t_runName	The run name when used with a run plan of the implementation. This is optional.

Value Returned

l_impTests	The list of all the tests of the specified cellview history.
`nil`	The specified implementation or run does not exist or there is no test in the specified implementation or run.

Examples

The following example opens a Verifier cellview and retrieves all the tests belonging to the specified implementation.

```
uid=verifOpenCellView("test" "sample" "verifier")
=> 0
```

```
tests=verifGetImpTests(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")
=> ("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
```

For an example on how `verifGetImpTests` can be used in a custom function, see Copy Implementation Units to Requirements.

Related Topics

verifGetImpTestOutputs

Implementation Functions

verifGetMappableType

```
verifGetMappableType(
     g_sessionId
     t_path
     )
     => t_type / nil
```

Description

Returns the type of the mappable object.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_path	Hierarchical path and name of the mappable object. Each level of the hierarchy is separated by a '/'.

Value Returned

t_type	The mappable object type. The type can be one of these: `Implementation`, `Test`, `Output`, `Run`, `StatOutput`, and `StatTest`.
`nil`	The operation was unsuccessful.

Examples

The following set of examples show how to retrieve the mappable object from a cellview in different scenarios.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active")
=> "Implementation"
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/
NoSuchTest")
=> nil
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/
opamp090:full_diff_opamp_AC:1")
=> "Test"
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/
opamp090:full_diff_opamp_AC:1/Current")
=> "Output"
```

Related Topics

verifAddCustomOutput

verifAddImp

Implementation Functions

verifGetTypicalSetup

```
verifGetTypicalSetup(
     g_sessionId
     )
     => l_typicalSetup / nil
```

Description

Retrieves the typical setup from a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_typicalSetup	Disembodied property list (DPL) containing the typical setup of the specified session.
`nil`	There is no typical setup or there are some failures.

Examples

The following example starts a Verifier session with a cellview and then retrievs the typical setup.

```
sessionId = verifOpenCellView("amsPLL" "TOP_verification" "verification"
=> 0
verifGetTypicalSetup(sessionId)
=> (nil vdd "1.5")
```

Related Topics

verifGetImpData

verifSetImpSetPreRunScript

Implementation Functions

verifMoveImp

```
verifMoveImp(
     g_sessionId
     t_lib
     t_cell
     t_view
     t_history
     [ x_itemIndex ]
     )
     => t / nil
```

Description

Moves an implementation to the specified location in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_lib	The library name of the implementation.
t_cell	The cell name of the implementation.
t_view	The view name of the implementation.
t_history	The history name of the implementation.
x_itemIndex	The item index of the implementation in the implementations list. If you do not specify the index, its default value `0` is used, which moves the specified implementation as the last entry.

Value Returned

`t`	The implementation is moved to the specified location.
`nil`	The implementation is not moved to the specified location.

Examples

The following example opens a Verifier cellview and lists the implementation cellviews. It moves the last implementation cellview to the specified location.

```
uid = verifOpenCellView("test" "sample" "verifier")
```

```
=> 0
```

Moves the implementation to be the last item.

```
verifMoveImp(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")
=> t
```

Moves the implementation to be the first item.

```
verifMoveImp(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active" 1)
```

Related Topics

verifAddCustomOutput

verifAddImp

verifMoveImp

verifOverwriteSpec

verifRemoveImp

verifSetImpData

Implementation Functions

verifOverwriteSpec

```
verifOverwriteSpec(
     g_sessionId
     g_impLib
     g_impCell
     g_impView
     g_impHistory
     [ ?testName t_testName ]
     [ ?outputName t_outputName ]
     [ ?pushToImp g_pushToImp ]
     )
     => t / nil
```

Description

Pulls or pushes the specification and unit from to the mapped implementation items hierarchically.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	Library name of the implementation.
t_impCell	Cell name of the implementation.
t_impView	View name of the implementation.
t_impHistory	History name of the implementation.
`?testName` *t_testName*	Name of the implementation test.
`?outputName` *t_outputName*	Name of the implementation output.
`?pushToImp` *g_pushToImp*	If enabled, pushes the requirement's specification and unit to mapped implementation items. This is equivalent to 'Overwrite Implementation Specification'. Default is *t*.

Value Returned

`t`	Successfully overwrites the specification and unit of the requirement or implementation.
`nil`	Failed to overwrite the specification and unit of the requirement or implementation.

Examples

The following example opens a Verifier cellview that has an implementation `Two_Stage_Opamp/OpAmp/maestro_nominal/Active`.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Overwrites the specification and unit for outputs belongs to the implementation.

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=> t
```

Overwrites the specification and unit for outputs belongs to the test.

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?testName "AC")
=> t
```

Overwrites the specification and unit for output 'Op_Region' belongs to the test `AC`.

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?testName "AC" ?outputName "Op_Region")
=> t
```

Overwrites the specification and unit for requirements that are mapped to outputs belongs to the implementation.

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?pushToImp nil)
=> t
```

Related Topics

verifAddImp

verifMoveImp

verifOverwriteSpec

verifRemoveImp

verifRemoveImp

```
verifRemoveImp(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     )
     => t / nil
```

Description

Deletes an implementation from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	The library name of the implementation.
t_impCell	The cell name of the implementation.
t_impView	The view name of the implementation.
t_impHistory	The history name of the implementation.

Value Returned

`t`	The specified implementation is deleted from the Verifier session.
`nil`	The specified implementation does not exist in the Verifier session for deletion, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation `Two_Stage_Opamp/OpAmp/maestro_nominal/Active`. This implementation is then deleted from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Gets all implementations.

```
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.RO")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active")
("amsPLL" "CP_TB" "maestro" "Active")
)
```

Deletes the specified implementation.

```
verifRemoveImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=> t
```

Related Topics

verifAddImp

verifMoveImp

Implementation Functions

verifRestart

```
verifRestart(
     g_sessionId
     [ ?implementation g_implementation ]
     [ ?impSet g_impSet ]
     )
     => t / nil
```

Description

Restarts the simulation for one or more implementations. If no implementation or implementation set is specified, all running implementations in the session are restarted.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *g_implementation*	The name of single implementation or a list of implementation names where each implementation name can be either a string such as `Lib/Cell/View/History` or a list such as `'("Lib" "Cell" "View" "History")`. If no implementation is specified, all running implementations in the session are restarted.
`?impSet` *g_impSet*	The name of a single implementation set or a list of names of implementation sets.

Value Returned

`t`	The specified implementation is restarted.
`nil`	The specified implementation fails to restart.

Examples

The following example starts a Verifier session with a cellview that has three implementations `Two_Stage_Opamp/OpAmp/maestro_nominal/Active`, `Two_Stage_Opamp/OpAmp/maestro_basic/Active`, and `Two_Stage_Opamp/OpAmp/maestro_run_plan/Active`. Here, the first two implementations exist in an implementation set `Weekly`.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Specifies the list of implementations to be run. When the implementation is running, the return value would be `t`.

```
verifRun(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp" "maestro_nominal"
"Active")))
Verifier: Started running implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Active' at 'Feb 16 09:31:05 2023'
=> t
```

Specifies the implementation sets to be run. When the implementation is running, the return value would be `t`.

```
verifRun(uid ?impSet "Weekly")
=> t
```

Restarts the specified implementations when both `Two_Stage_Opamp/OpAmp/maestro_nominal/Active` and `Two_Stage_Opamp/OpAmp/maestro_basic/Active` are in running state.

```
verifRestart(uid ?impSet "Weekly")
=>(t t)
```

Restarts the specified implementations when only `Two_Stage_Opamp/OpAmp/maestro_nominal/Active` is running and `Two_Stage_Opamp/OpAmp/maestro_basic/Active` is not running.

```
verifRestart(uid ?impSet "Weekly")
=>(t "")
```

Restarts the specified list of implementations when all three implementations are in running state

```
verifRestart(uid ?impSet "Weekly" ?implementation list(list("Two_Stage_Opamp"
"OpAmp" "maestro_nominal" "Active")))
=>(t t t)
```

Related Topics

Restarting Implementation Jobs

Implementation Functions

verifSetImpData

```
verifSetImpData(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     t_historyName
     g_data
     [ ?runName t_runName ]
     [ ?relName t_relName ]
     [ ?testName t_testName ]
     [ ?outputName t_outputName ]
     [ ?statName t_statName ]
     )
     => t / nil
```

Description

Sets the specified data for a given implementation item.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	Library name of the existing implementation.
t_cellName	Cell name of the existing implementation.
t_viewName	View name of the existing implementation.
t_historyName	History name of the existing implementation.
g_data	List of key or value pairs that must be set on the resulting output.
`?runName` *t_runName*	Run name in the existing implementation.
`?relName` *t_relName*	Reliability name in the existing implementation.
`?testName` *t_testName*	Test name in the existing implementation.
`?outputName` *t_outputName*	Output name in the existing implementation.

`?statName` *`t_statName`*	Statistical output name in the existing implementation.

Value Returned

`t`	Specified data was successfully set for the item.
`nil`	Specified data was not set because of an error.

Examples

The following example shows how to set the data for an implementation item.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Changes the specification and units for the output.

```
verifSetImpData(sess "lib" "cell" "view" "Active" '("spec" "> 45" "unit" "V")
?testName "Test1" ?outputName "Gain")
```

Displays an error because the implementation name cannot be changed.

```
verifSetImpData(sess "lib" "cell" "view" "Active" '("name" "abc"))
```

Sets the custom implementation data for an exiting output.

```
verifSetImpData(sess "l" "c" "v" "h" list("details" "45" "unit" "V" "spec" "> 45")
?testName "test" ?outputName "output")
```

Related Topics

verifGetImpData

Implementation Functions

verifSetImpEstRunTime

```
verifSetImpEstRunTime(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     x_estRunTime
     )
     => t / nil
```

Description

Sets the `EstRunTime` of specified implementation in a Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_impLib`	Library name of the implementation.
`t_impCell`	Cell name of the implementation.
`t_impView`	View name of the implementation.
`t_impHistory`	History name of the implementation.
`x_estRunTime`	Integer number of seconds for estimated run time.

Value Returned

`t`	`EstRunTime` of specified implementation is set to `x_estRunTime` successfully.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `EstRunTime` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0
```

```
verifSetImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active" 70)
=>t
```

Related Topics

verifGetImpEstRunTime

verifUpdateImpEstRunTime

Implementation Functions

verifSetImpPriority

```
verifSetImpPriority(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     x_priority
     )
     => t / nil
```

Description

Sets the `Priority` of the specified implementation in a Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_impLib`	Library name of the implementation.
`t_impCell`	Cell name of the implementation.
`t_impView`	View name of the implementation.
`t_impHistory`	History name of the implementation.
`x_priority`	Integer number for priority.

Value Returned

`t`	`Priority` of specified implementation is set to `x_priority` successfully.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `Priority` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0
```

```
verifSetImpPriority(1 "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active" 70)
=>t
```

Related Topics

verifGetImpPriority

Implementation Functions

verifSetImpSetPreRunScript

```
verifSetImpSetPreRunScript(
     g_sessionId
     t_impSetName
     t_preRunScriptFileName
     )
     => t / nil
```

Description

Sets the specified pre-run script file name for an implementation set.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	Name of the implementation set.
t_preRunScriptFileName	File name of the pre-run script attached to an implementation set.

Value Returned

`t`	The pre-run script for the given implementation set is specified successfully.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the pre-run script for an implementation set.

```
uid = verifOpenCellView("test" "pre-run" "verifier")
=>0
verifSetImpSetPreRunScript(uid "Weekly" "prerun.il")
=>t
```

Returns `nil` if the pre-run script file does not exist.

```
verifSetImpSetPreRunScript(uid "Weekly" "abc.il")
*WARNING* (VERIFIER-1368): The pre-run script file 'abc.il' for implementation set
'Weekly' does not exist.
Either update the path to the script, or ensure that the script exists in the
location before running this implementation set.
=>nil
```

Related Topics

verifSetImpSetPreRunScript

Implementation Functions

verifUpdate

```
verifUpdate(
     g_sessionId
     )
     => t / nil
```

Description

Updates the Verifier setup from local implementations, and referenced Verifier cellviews.

Arguments

`t_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Verifier setup was updated successfully.
`nil`	Verifier setup failed to update.

Examples

The following example opens a Verifier cellview, updates the Verifier setup from local implementations, and referenced Verifier cellviews.

```
sessionId = verifOpenCellView("test" "sample" "verifier")
=> 0
verifUpdate(sessionId)
=> t
```

Related Topics

verifOverwriteSpec

verifRemoveImp

verifSetImpData

verifSetImpPriority

verifSetImpSetPreRunScript

verifUpdateImpEstRunTime

Implementation Functions

verifUpdateImpEstRunTime

```
verifUpdateImpEstRunTime(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     )
     => t / nil
```

Description

Updates the `EstRunTime` to be the actual recorded run time from run summary data.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	Library name of the implementation.
t_impCell	Cell name of the implementation.
t_impView	View name of the implementation.
t_impHistory	History name of the implementation.
x_priority	Integer number for priority.

Value Returned

`t`	`EstRunTime` of specified implementation is updated successfully.
`nil`	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `Priority` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0
```

```
verifUpdateImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=>t
```

Related Topics

verifGetImpEstRunTime

verifSetImpEstRunTime

Implementation Functions

4

Requirement-to-Implementation Mapping Functions

You can map implementation cellviews, tests, and outputs to their corresponding requirements of the type *Spec Pass*, *Ran OK*, and *Manual*. If the implementation cellview uses the *Monte Carlo Sampling* run mode, you can map that cellview, its tests, output values, and statistical values with requirements.

It is possible to map multiple implementations with multiple requirements. If you want to map multiple requirements with an implementation, ensure that the requirements have the same specification.

Verifier uses the mapping, along with simulation results and preference settings, to determine the verification status of requirements. The correctness of the overall verification status depends on the completeness of the verification plan and mappings.

The following SKILL functions let you manage the mappings between requirements and implementations in the ADE Verifier environment:

- verifExportMapping
- verifGetImpMapping
- verifGetReqMapping
- verifImportMapping
- verifMapping

Related Topics

Requirement Functions

Implementation Functions

verifExportMapping

```
verifExportMapping(
     g_sessionId
     t_fileName
     [ S_type ]
     [ g_confirmOverwrite ]
     )
     => t / nil
```

Description

Export the mapping to either a CSV or Excel file.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	The name of the file to export the mapping to.
S_type	Symbol or string to identify the export type. The supported types are "Excel" or "CSV". The default value is "CSV". The string name is case insensitive.
g_confirmOverwrite	If `t` and the file already exists, will ask for confirmation to overwrite it, otherwise will just go ahead and overwrite it.

Value Returned

`t`	Exported the mapping.
`nil`	Failed to export the mapping.

Examples

Opens a Verifier cellview and exports the mapping information of all implementations to the specified file format.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 1
verifExportMapping(sess "mapping.csv")
=> t
verifExportMapping(sess "mapping.xlsx" "Excel" t)
=> t
```

Related Topics

verifGetImpMapping

verifGetReqMapping

verifImportMapping

verifMapping

Requirement-to-Implementation Mapping Functions

verifGetImpMapping

```
verifGetImpMapping(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     t_historyName
     [ ?runName t_runName ]
     [ ?testName t_testName ]
     [ ?outputName t_outputName
     [ ?statName t_statName ]
     )
     => l_reqIds / nil
```

Description

Returns the list of requirements that are mapped to the specified mappable object. The mappable object could be an implementation cellview, run, test or output.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	Name of the library that contains the implementation.
t_cellName	Name of the cell that contains the implementation.
t_viewName	Name of the view that contains the implementation.
t_historyName	Name of the history in the implementation.
`?runName` *t_runName*	Name of the implementation run.
`?testName` *t_testName*	Name of the test.
`?outputName` *t_outputName*	Name of the output.
`?statName` *t_statName*	Name of the statistical test or output.

Value Returned

`l_reqIds`	The list of requirements that are mapped to the specified mappable object.
`nil`	The specified mappable object does not exist or is not mapped.

Examples

Opens a Verifier cellview and gets a list of requirement IDs mapped to the statistical output `AC::Overall_Yield`.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifGetImpMapping(0 "test" "setup" "verifier" "Active" ?testName "AC" ?statName
"AC::Overall_Yield")
=> ("ID_1" "ID_5.1")
```

Related Topics

verifExportMapping

verifGetReqMapping

verifImportMapping

verifMapping

Requirement-to-Implementation Mapping Functions

verifGetReqMapping

```
verifGetReqMapping(
     g_sessionId
     t_reqId
     [ ?types l_types ]
     => l_implementations / nil
```

Description

Returns a list of mappable objects mapped to the specified requirement. The mappable object could be an implementation cellview, run, test or output.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID of the requirement.
`?types` *l_types*	The mappable object types to be returned. Valid values are "implementation", "test", "output" and "run". If a value is not specified, all types are returned.

Value Returned

l_implementations	Successful operation.
`nil`	Unsuccessful operation.

Examples

Opens a Verifier cellview and gets a list of all implementations mapped to requirement "ID_1" and then gets a list of outputs mapped to the requirement.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifGetReqMapping(sessionId "ID_1")
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "TRAN"
    "PhaseMargin"
)
("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "TRAN")
("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")
)
verifGetReqMapping(sessionId "ID_1" ?types '("output"))(("Two_Stage_Opamp" "OpAmp"
"maestro_MC" "Active" "TRAN"
```

```
    "PhaseMargin"
))
```

Related Topics

verifExportMapping

verifGetImpMapping

verifImportMapping

verifMapping

Requirement-to-Implementation Mapping Functions

verifImportMapping

```
verifImportMapping(
     g_sessionId
     t_fileName
     S_type
     )
     => t / nil
```

Description

Import mapping from an Excel or CSV file.The first row of the file will be ignored as it should be a header row. The rest of the rows should have at least two columns: `RequirementId` and `MappingName`. Any other columns will be ignored. Each row is a single requirement-to-implementation mapping, so if a requirement is mapped to multiple implementations, then there will be multiple rows with the same requirement ID in the first column.

For example:

```
Requirement,Mapping
ID6,opamp090/full_diff_opamp_AC/maestro/Active/opamp090:full_diff_opamp_TRAN:1/
SlewRate
ID6,opamp090/full_diff_opamp_AC_bad/maestro/Active/
opamp090:full_diff_opamp_TRAN:1/SlewRate
ID5,opamp090/full_diff_opamp_AC/maestro/Active/opamp090:full_diff_opamp_TRAN:1/
SettlingTime
```

Shell-style wildcard character "*" can be used in the names, but it is recommended to avoid numerous matches when importing the file. For example:

```
ID*,op*/full_diff_opamp_AC/maestro*/Active
```

This mapping is valid, but would map every requirement beginning with ID to all implementations that match. Any existing mapping on the requirement is retained, so any new mapping is added on top of that.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	File to import the mapping from.
S_type	String or symbol to specify the type of file. It can be either `"CSV"` (the default) or `"Excel"`.

Value Returned

`t`	The import was successful.
`nil`	The import was unsuccessful.

Examples

Opens a Verifier cellview and imports the mapping information from the specified file.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifExportMapping(sess "map.csv")
=> t
sess2 = verifOpenCellView("test" "sample2" "verifier")
=> 1
verifImportMapping(sess2 "map.csv")
=> t
```

Related Topics

verifExportMapping

verifGetImpMapping

verifGetReqMapping

verifMapping

Requirement-to-Implementation Mapping Functions

verifMapping

```
verifMapping(
     g_sessionId
     g_reqsList
     g_impsList
     [ g_operation ]
     )
     => t / nil
```

Description

Maps one or more requirements to one or more implementations in a Verifier session. Each of the requirement map objects can include the cellview, run, test, and output information. This function is also used to modify or delete the existing requirement-implementation mappings. For example, if the requirements list is empty, all the mapping information for the specified implementations is deleted.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_req	Single requirement ID or list of requirement IDs to be mapped. The requirements can be nil, a string, or a list of strings.
l_impsList	The implementations to be mapped to the specified requirements. The implementations can be nil, or a string, a list of strings, or a list of lists of strings.

`g_operation`	Specifies whether to merge the specified mapping with any existing mapping, or to overwrite any existing mapping.

The following string values are used to specify the task to be performed for outdated mapping information:

- `merge`: Merges the old mapping information. Similar to the behavior of *Include Old Mappings* on the *Requirements or Implementations Mapped but Not Selected* dialog box.
- `overwrite` (default value): Removes the old mapping information and maps only to the new implementations; Similar to the behavior of *Continue* on the *Requirements or Implementations Mapped but Not Selected* dialog box.

An implementation name is specified as a list of strings or as string with each item separated by "/" from the following components:

```
<lib> <cell> <view> <history> [<run>] [<test>
[<statisticalTest>] || [<output> [<statisticalOutput>]]]
```

For example:

```
"lib/cell/view/history"
list("lib" "cell" "view" "history")
"lib/cell/view/history/run"
"lib/cell/view/history/run/test"
"lib/cell/view/history/test/statTest"
"lib/cell/view/history/run/test/statTest"
"lib/cell/view/history/test/output"
"lib/cell/view/history/test/output/statOut"
```

Value Returned

`t`	The specified requirements have been mapped to the specified implementations.
`nil`	The specified requirements and implementations could not be mapped.

Examples

The following example opens a Verifier cellview and maps the specified requirements to the specified implementations. It also illustrates how to delete the mapping of requirements and implementations using `verifMapping`.

Maps the requirement with ID `"ID_LT8XzV"` to implementation "l/c/v/h".

```
sess = verifOpenCellView("tests" "sample" "verifier")
verifMapping(sess "ID_LT8XzV" "l/c/v/h")
=> t
```

Removes the mapping of requirement ID `"ID_LT8XzV"`.

```
verifMapping(sess "ID_LT8XzV" nil)
=> t
```

Removes the mapping of implementation "l/c/v/h".

```
verifMapping(sess nil list("l" "c" "v" "h"))
=> t
```

Maps both requirement IDs "`"ID_LT8XzV"` and `"ID_LT6YzV"`" to the same output.

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list("l" "c" "v" "h" "test"
"output1"))
=> t
```

Maps both requirement IDs "`"ID_LT8XzV"` and `"ID_LT6YzV"`" to the same two implementations.

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2")))
=> t
```

Adds the mapping of both requirement IDs "`"ID_LT8XzV"` and `"ID_LT6YzV"`" to the same two implementations to any existing mapping.

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2")) "merge")
=> t
```

Specifies the implementation using the list notation.

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")))
=> t
```

Specifies two implementations using the list notation.

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2")))
=> t
```

Related Topics

verifExportMapping

verifGetImpMapping

verifGetReqMapping

verifImportMapping

Requirement-to-Implementation Mapping Functions

5

Simulation and Results Extraction Functions

Verifier uses the simulation results of the implementation cellviews to determine the verification status of the entire project, and for each requirement in the project. To capture the results, you can run the implementation cellviews from Verifier or load the existing run results in Verifier. By organizing implementation cellviews into implementation sets, you can run all the cellviews in a set.

The following SKILL functions let you manage the simulations and result extraction in the ADE Verifier environment:

- Implementation Set Functions - These functions let you manage implementation sets.
  - verifAddImpSet
  - verifAddImpToImpSet
  - verifGetImpSets
  - verifGetImpsInImpSet
  - verifSetImpSetName
  - verifRemoveImpSet
  - verifRemoveImpFromImpSet
- Results Extraction Functions - These functions let you retrieve simulation results.
  - verifGetResultDataForImp
  - verifGetResultDataForReq
  - verifPublishHTML
  - verifReloadAllRes

- Simulation Functions - Use the following functions to manage simulations.
  - verifCheck
  - verifCheckImp
  - verifCreateRunSummaryData
  - verifDeleteRunSummaryData
  - verifImpIsRun
  - verifRun
  - verifSetImpRun
  - verifStop

Related Topics

Implementation Functions

verifAddImpSet

```
verifAddImpSet(
     g_sessionId
     t_impSetName
     )
     => t / nil
```

Description

Adds a new implementation set in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	Name of implementation set.

Value Returned

`t`	The implementation set is added to the specified Verifier session.
`nil`	The implementation set is not added to the specified Verifier session because of errors.

Examples

The following example opens a Verifier cellview and adds a new implementation set.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 1
```

```
verifAddImpSet(sessionId "Weekly")
=> t
```

Related Topics

verifAddImpToImpSet

verifGetImpSets

verifGetImpsInImpSet

verifAddImpToImpSet

```
verifAddImpToImpSet(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     t_impSetName
     )
     => t / nil
```

Description

Adds an implementation to the specified implementation set in a Verifier session. You can add an implementation to multiple implementation sets.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	The library name of the implementation.
t_impCell	The cell name of the implementation.
t_impView	The view name of the implementation.
t_impHistory	The history name of the implementation.
t_impSetName	The name of the implementation set.

Value Returned

`t`	The implementation is added to the implementation set.
`nilf`	The implementation or implementation set does not exist, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation '`opamp090/full_diff_opamp_AC/maestro/Active`'. It adds an implementation set 'Weekly' and then add this implementation to that set.

```
uid=verifOpenCellView("test" "sample" "verifier")
=> "0"
verifAddImpSet(uid "Weekly")
=> t
verifAddImpToImpSet(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active"
"Weekly")
=> t
```

Related Topics

verifAddImpSet

verifGetImpSets

verifGetImpsInImpSet

verifRemoveImpSet

verifRemoveImpFromImpSet

verifSetImpSetName

Simulation and Results Extraction Functions

verifCheck

```
verifCheck(
     g_sessionId
     [ ?implementation g_implementation ]
     [ ?impSet g_impSet ]
     )
     => t / nil
```

Description

Checks whether implementations changed since last run or reload by Verifier. In addition, it checks whether the lifetime of implementation simulation results is within the *Expiration Period of Implementation Results* since the last simulation of implementation cellview by Verifier or the last external creation of results. If no implementation or implementation set is specified, then all implementations in the session are checked.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *`g_implementation`*	Name of single implementation or list of implementation names where each implementation name can be either a string such as `Lib/Cell/View/History` or a list `'("Lib" "Cell" "View" "History")`. If no implementation is specified, all implementations in the session are run.
`?impSet` *`g_impSet`*	Name of a single implementation set or a list of implementation set names.

Value Returned

`t`	Check for changes in implementations and simulation results is done.
`nil`	Check for changes in implementation and simulation results is not done.

Examples

The following example starts a Verifier session with a cellview and performs the check.

```
uid = verifOpenCellView("test" "results" "verifier")
=>0
```

Gets all implementations.

```
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.RO")
    ("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Interactive.0")
)
```

Does the check.

```
verifCheck(uid)
INFO (VERIFIER-1329): The implementation 'Two_Stage_Opamp/OpAmp/maestro_MC/
MonteCarlo.3.RO' has not yet been run by Verifier.
Therefore, no checks can be performed.

INFO (VERIFIER-1324): The implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Interactive.0' has changed as below:
some elements in the following test state have changed:
'AC: environmentOptions, faultFiles, faultRules, faults, outputs'
'TRAN: analyses, environmentOptions, faultFiles, faultRules, faults, outputs'
```

```
*WARNING* (VERIFIER-1326): Found outdated simulation results for implementation
'Two_Stage_Opamp/OpAmp/maestro_nominal/Interactive.0' because the implementation
cellview has changed since the last simulation run.
To ensure that Verifier has the latest simulation results, enable the preference
option 'Report identical history before run' and rerun the implementation from
Verifier.
=> t
```

Related Topics

verifCheckImp

Simulation and Results Extraction Functions

verifCheckImp

```
verifCheckImp(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     )
     => t / nil
```

Description

Checks for changes of the specified implementation and loaded results.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	Library name of the implementation.
t_impCell	Cell name of the implementation.
t_impView	View name of the implementation.
t_impHistory	History name of the implementation.

Value Returned

t	Check for changes is successful.
`nil`	Implementation does not exist, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview that has implementations `'opamp090/full_diff_opamp_AC/maestro/Active'` and `'Two_Stage_Opamp/OpAmp/maestro_basic/Active'`. It successfully checks for changes in implementations and loaded results.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0

verifCheckImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
```

```
=> INFO (VERIFIER-1329): The implementation 'Two_Stage_Opamp/OpAmp/
maestro_nominal/Active' has not yet been run by Verifier.
Therefore, no checks can be performed.
=> t
verifCheckImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_basic" "Active")
=> INFO (VERIFIER-1324): The implementation 'Two_Stage_Opamp/OpAmp/maestro_basic/
Active' has changed as below:
following elements in the setup have changed:
specifications

=> *WARNING* (VERIFIER-1326): Found outdated simulation results for implementation
'Two_Stage_Opamp/OpAmp/maestro_basic/Active' because the implementation cellview
has changed since the last simulation run.
To ensure that Verifier has the latest simulation results, enable the preference
option 'Report identical history before run' and rerun the implementation from
Verifier.
=> t
```

Related Topics

verifCheck

Simulation and Results Extraction Functions

verifCreateRunSummaryData

```
verifCreateRunSummaryData(
     g_sessionId
     [ ?implementation g_implementation ]
     )
     => t / nil
```

Description

Creates the run summary data for one or more implementations. If no implementation is specified, the run summary data is created for all implementations in the session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *`g_implementation`*	Name of an implementation or a list of implementation names where each implementation name can be either a string such as `Lib/Cell/View/History` or a list `'("Lib" "Cell" "View" "History")`. If no implementation is specified, run summary data is created for all implementations in the session.

Value Returned

`t`	Run summary data is created.
`nil`	Run summary data was not created for one or more specified implementations.

Examples

The following example starts a Verifier session with a cellview that has two implementations `Two_Stage_Opamp/OpAmp/maestro_MC/Active` and `Two_Stage_Opamp/OpAmp/maestro_nominal/Active`.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Creates and loads the run summary data files for the specified implementation.

```
verifCreateRunSummaryData(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active") ))
Verifier: Created and loaded the run summary data files for implementation
'Two_Stage_Opamp/OpAmp/maestro_nominal/Active'
=> t
```

Creates and loads the run summary data files for both implementations.

```
verifCreateRunSummaryData(uid)

Verifier: Created and loaded the run summary data files for both 'Two_Stage_Opamp/
OpAmp/maestro_MC/Active' and 'Two_Stage_Opamp/OpAmp/maestro_nominal/Active'

=>t
```

Related Topics

verifDeleteRunSummaryData

Simulation and Results Extraction Functions

verifDeleteRunSummaryData

```
verifDeleteRunSummaryData(
     g_sessionId
     [ ?implementation g_implementation ]
     )
     => t / nil
```

Description

Deletes run summary data for one or more implementations. If no implementation is specified, the run summary data is deleted for all implementations in the session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *g_implementation*	Name of an implementation or a list of implementation names where each implementation name can be either a string such as `Lib/Cell/View/History` or a list `'("Lib" "Cell" "View" "History")`. If no implementation is specified, run summary data is deleted for all implementations in the session.

Value Returned

`t`	Run summary data is deleted.
`nil`	Run summary data was not deleted for one or more specified implementations.

Examples

The following example starts a Verifier session with a cellview that has two implementations `Two_Stage_Opamp/OpAmp/maestro_MC/Active` and `Two_Stage_Opamp/OpAmp/maestro_nominal/Active`.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Deletes the run summary data files for the specified implementation.

```
verifDeleteRunSummaryData(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active") ))

Verifier: Delete the run summary data files for implementation 'Two_Stage_Opamp/
OpAmp/maestro_nomimal/Active'
=> t
```

Deletes the run summary data files for both implementations.

```
verifDeleteRunSummaryData(uid)

Verifier: Delete the run summary data files for both 'Two_Stage_Opamp/OpAmp/
maestro_MC/Active' and 'Two_Stage_Opamp/OpAmp/maestro_nominal/Active'

=>t
```

Related Topics

verifCreateRunSummaryData

Simulation and Results Extraction Functions

verifCopyAndUpdateResultsFromUserDefinedDirectory

```
verifCopyAndUpdateResultsFromUserDefinedDirectory(
     g_sessionId
     )
     => t / nil
```

Description

Copies the run summary data files from the user-defined directory to the `results` directory of the current cellview. Additionally, updates the current session setup to restore the run summary data file into the `results` directory of the current cellview.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Copies the run summary data files from the user-defined directory to the `results` directory of the current cellview and updates the current setup successfully.
`nil`	The command is not successful.

Examples

The following example shows how to use this function.

```
uid = verifOpenCellView("test" "results" "verifier" ?openWindow nil)

INFO (VERIFIER-6603): You have opened a cellview that uses the user-defined
directory './Verifier_Results', earlier available from Edit -> Preference-> Run -
> Location of the Implementation Run Summary Data.
ADE Verifier no longer supports this feature and will automatically modify this
setup to store the run summary data in the 'results' directory of the current
cellview.
To create the run summary data, run simulations or load the implementation history
results.
To copy the previous results from './Verifier_Results' and update the current
setup, make the session editable and use the SKILL function
'verifCopyAndUpdateResultsFromUserDefinedDirectory(sessionId)'.
To continue using the user-defined location for the run summary data, contact
Cadence Customer Support for assistance.
=> 0
```

```
verifGetOptionVal(uid "palswhere")
=>"userdir"

verifCopyAndUpdateResultsFromUserDefinedDirectory(uid)

INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_basic_Active.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.Nominal.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.MC.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.Run.0.xml.
=> t

verifGetOptionVal(uid "palswhere")
=> "currentcellview"

verifCloseSession(uid ?saveIfModified t) INFO (VERIFIER-1507): Saved cellview
test.ro:verifier_user
=> t
```

Related Topics

verifEvaluateResults

Simulation and Results Extraction Functions

verifEvaluateResults

```
verifEvaluateResults(
     g_sessionId
     )
     => t / nil
```

Description

Evaluates all requirements and does a force update of the results in special cases.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Re-evaluates the results for all requirements.
`nil`	The command is not successful.

Examples

The following example shows how to use this function.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifEvaluateResults(uid)
=> t
```

Related Topics

verifCopyAndUpdateResultsFromUserDefinedDirectory

Simulation and Results Extraction Functions

verifGetImpSets

```
verifGetImpSets(
     g_sessionId
     [ ?parentName t_parentName ]
     [ ?runnable g_runnable ]
     [ ?isGroup g_isGroup ]
     )
     => l_impSets / nil
```

Description

Retrieves the list of all the implementation sets in a Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?parentName` *`t_parentName`*	Name of the parent implementation set.
`?runnable` *`g_runnable`*	A check that when enabled, retrieves the implementation set that can be run from Verifier. Only the top-level implementation set is runnable.
`?isGroup` *`g_isGroup`*	A check that when enabled, retrieves the implementation set which is a group of implementations.

Value Returned

`l_impSets`	The list of implementation sets in the Verifier session.
`nil`	There is no implementation set in the Verifier session, or the command is not successful.

Examples

The following example opens a Verifier cellview and retrieves all the implementation sets.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetImpSets(uid)
=> ("Weekly" "Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active" "Weekly->PlanB"
"Daily")
```

Retrieves the list of dependent implementation sets from a specified implementation or implementation set.

```
verifGetImpSets(sess ?parentName "Weekly")
=> ("Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active" "Weekly->PlanB")
```

Retrieves the list of top-level implementation sets. Only a top-level implementation set is runnable.

```
verifGetImpSets(sess ?runnable t) nil)
=> ("Weekly" "Daily")
```

Retrieves the list of implementation set that include a group.

```
verifGetImpSets(sess ?isGroup t)
=> ("Weekly" "Weekly->PlanB" "Daily")
```

Retrieves the list of implementation sets that include an implementation.

```
verifGetImpSets(sess ?isGroup nil)
=> ("Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active")
```

Related Topics

verifAddImpSet

verifAddImpToImpSet

verifGetImpsInImpSet

verifRemoveImpSet

verifRemoveImpFromImpSet

verifSetImpSetName

Simulation and Results Extraction Functions

verifGetImpsInImpSet

```
verifGetImpsInImpSet(
     g_sessionId
     t_impSetName
     )
     => l_impSets / nil
```

Description

Retrieves the list of implementations from the specified implementation set in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	The name of the implementation set.

Value Returned

l_impSets	The list of implementations from the specified implementation set in a Verifier session.
`nil`	The implementation set does not exist or there is no implementation in the specified set.

Examples

The following example opens a Verifier cellview and retrieves the list of implementations from the specified implementation set in a Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetImpSets(uid)
("Weekly" "Daily")
verifGetImpsInImpSet(uid "Weekly")
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.RO")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Interactive.0")
)
verifGetImpsInImpSet(uid "Daily")
=> nil
```

Related Topics

verifAddImpSet

verifAddImpToImpSet

verifGetImpSets

verifRemoveImpSet

verifRemoveImpFromImpSet

verifSetImpSetName

Simulation and Results Extraction Functions

verifGetResultDataForImp

```
verifGetResultDataForImp(
     g_sessionId
     t_lib
     t_cell
     t_view
     t_history
     [ ?runName t_runName ]
     [ ?testName t_testName ]
     [ ?outputName t_outputName ]
     [ ?statName t_statName ]
     )
     => o_resultDataTable / nil
```

Description

Returns all the result data for the specified implementation in a Verifier session. For each sub tables, there is a table that contains the multiple child items result data.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_lib	The library name of the implementation.
t_cell	The cell name of the implementation.
t_view	The view name of the implementation.
t_history	The history name of the implementation.
`?runName` *t_runName*	The name for the implementation in 'Run Plan' mode. This argument is optional.
`?testName` *t_testName*	The name for the implementation test. This argument is optional.
`?outputName` *t_outputName*	The name for the implementation output. This argument is optional.
`?statName` *t_statName*	The name of the statistical value for implementation test or output. This argument is optional.

Value Returned

o_resultDataTable	The table containing the result data for all the child items of the specified implementation.
`nil`	The specified implementation does not exist, the results data is not available, or the command is not successful.

Examples

The following example opens a Verifier cellview, and retrieves the result data of the specified implementation.

```
uid = verifOpenCellView("test" "results" "verifier")
=>0
tableToList(verifGetResultDataForImp(uid "Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active"))
(("Gain" table:info)
    ("TRAN" table:info)
    ("SettlingTime" table:info)
    ("Current" table:info)
    ("Voffset" table:info)
    ("RelativeSwingPercent" table:info)
    ("PhaseMargin" table:info)
    ("UGF" table:info)
    ("Implementation" list("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"))
    ("Swing" table:info)
    ("AC" table:info)
    ("Op_Region" table:info)
)
```

To use the result data sub table:

```
tableToList(verifGetResultDataForImp(0 "Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active")["Current"])
(("points" "1")
      ("history" "verifier_run.102.RO")
      ("spec" "tol 7m 6%")
      ("disabledPoints" "0")
      ("min" "7.13069521418212e-3")
      ("max" "7.13069521418212e-3")
      ("canceledPoints" "0")
      ("Implementation" list("opamp090" "full_diff_opamp_AC" "maestro" "Active"))
      ("failedPoints" "0")
      ("minParams" "Model=(gpdk090.scs,MC_models),corner=C1")
      ("Test" "opamp090:full_diff_opamp_AC:1")
      ("Output" "Current")
      ("status" "Pass")
      ("maxParams" "Model=(gpdk090.scs,MC_models),corner=C1")
      ("passedPoints" "1")
)
```

Related Topics

verifGetResultDataForReq

verifPublishHTML

verifReloadAllRes

Simulation and Results Extraction Functions

verifGetResultDataForReq

```
verifGetResultDataForReq(
     g_sessionId
     t_reqId
     )
     => o_resultDataTable / nil
```

Description

Returns the simulation result data for the specified requirement in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_reqId	The ID of the requirement.

Value Returned

o_resultDataTable	The table containing the result data.
`nil`	The specified requirement does not exist, the result data is not available, or the command is not successful.

Examples

The following example opens a Verifier cellview and retrieves the result data of the implementation mapped to the requirement in that cellview.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Requirement 'ID1' of 'Note' type

```
tableToList(verifGetResultDataForReq(uid "ID1"))
(("reqStatus" "0%")
     ("passed" "0")
     ("reqTitle" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active (Note)"))
     ("reqType" "Note")
     ("noResults" "77")
     ("reqId" "ID1")
     ("reqHier" "1")
```

```
    ("failed" "0")
    ("unmapped" "0")
)
```

Mapped requirement 'ID1.1'

```
tableToList(verifGetResultDataForReq(uid "ID1.1"))
(("reqStatus" "Not Run")
    ("passed" "0")
    ("reqTitle" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"))
    ("reqType" "Ran Ok")
    ("noResults" "1")
    ("reqId" "ID1.1")
    ("reqHier" "1->1.1")
    ("failed" "0")
    (list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active") table:info)
)
```

Related Topics

verifGetResultDataForImp

verifPublishHTML

verifReloadAllRes

Simulation and Results Extraction Functions

verifImpIsRun

```
verifImpIsRun(
     g_sessionId
     t_libName
     t_cellName
     t_viewName
     t_historyName
     )
     => t / nil
```

Description

Checks if the specified implementation cellview can be run from a Verifier session.

You can load the simulation results from the selected history of an implementation cellview that cannot be run from Verifier.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_libName	The library name of the implementation.
t_cellName	The cell name of the implementation.
t_viewName	The view name of the implementation.
t_historyName	The history name of the implementation.

Value Returned

`t`	The implementation has the *Run* check box checked.
`nil`	The specified implementation does not exist or cannot be run from Verifier.

Examples

The following example shows how to use this function.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifImpIsRun(sess "test" "sample" "maestro" "Active")
=> t
```

Related Topics

verifRun

verifSetImpRun

verifCheck

verifCheckImp

verifStop

Simulation and Results Extraction Functions

verifPublishHTML

```
verifPublishHTML(
     g_sessionID
     g_openBrowser
     )
     => t / nil
```

Description

Generates and optionally displays an HTML report in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_openBrowser	An optional boolean argument. If this is not `nil`, the generated HTML report is displayed in the default web browser. Default value is `nil`.

Value Returned

`t`	The HTML file was generated.
`nil`	The HTML file was not generated.

Examples

The following example opens a Verifier cellview and generates an HTML report.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifPublishHTML(sessionId t)
=> t
```

Related Topics

verifGetResultDataForImp

verifGetResultDataForReq

verifReloadAllRes

Simulation and Results Extraction Functions

verifReloadAllRes

```
verifReloadAllRes(
     g_sessionId
     )
     => t / nil
```

Description

Reloads the simulation results for all implementations that have the *Run* check box unchecked. This is equivalent to clicking the *Reload Simulation Results* button on the Verifier toolbar.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	All implementation simulation results are reloaded.
`nil`	Failed to reload all the implementation simulation results.

Examples

This example shows how to reload the simulation results for all implementations within the Verifier session.

```
uid = verifOpenCellView("test" "sample" "verifier")
=> 0
verifReloadAllRes(uid)
=> t
```

Related Topics

verifGetResultDataForImp

verifGetResultDataForReq

verifPublishHTML

Simulation and Results Extraction Functions

verifRemoveImpFromImpSet

```
verifRemoveImpFromImpSet(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     t_impSetName
     )
     => t / nil
```

Description

Deletes an implementation from the specified implementation set in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	The library name of the implementation.
t_impCell	The cell name of the implementation.
t_impView	The view name of the implementation.
t_impHistory	The history name of the implementation.
t_impSetName	The name of the implementation set.

Value Returned

`t`	The specified implementation is deleted from the implementation set.
`nil`	The specified implementation set does not exist, the specified implementation cellview is not available in the implementation set, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation '`opamp090/full_diff_opamp_AC/maestro/Active`' and an implementation set 'Weekly'. It deletes this implementation from the specified implementation set.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifRemoveImpFromImpSet(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
"Weekly")
=> t
```

Related Topics

verifAddImpSet

verifAddImpToImpSet

verifGetImpSets

verifGetImpsInImpSet

verifRemoveImpSet

verifSetImpSetName

Simulation and Results Extraction Functions

verifRemoveImpSet

```
verifRemoveImpSet(
     g_sessionId
     t_impSetName
     )
     => t/nil
```

Description

Deletes the implementation set from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	The name of the implementation set.

Value Returned

`t`	The specified implementation set is deleted from the Verifier session.
`nil`	The specified implementation set does not exist, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation set 'Weekly'. It deletes this implementation set from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifRemoveImpSet(uid "Weekly")
=> t
```

Related Topics

verifAddImpSet

verifAddImpToImpSet

verifGetImpSets

verifRun

```
verifRun(
     g_sessionId
     [ ?implementation g_implementation ]
     [ ?impSet g_impset ]
     [ ?runPlanRun g_runPlanRun ]
     [ ?mode g_mode ]
     [ ?waitUntilDone g_waitUntilDone ]
     [ ?rerun g_rerun ]
     )
     => t / nil
```

Description

Runs or loads the result of the specified implementations in a Verifier session.

If the *Run* check box is enabled then it runs the implementation. If the check box is disabled, it loads any existing simulation results.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *g_implementation*	The name of single implementation or list of implementation names where each implementation name can be either a string like `"Lib/Cell/View/History"` or a list `'("Lib" "Cell" "View" "History")'`. When no implementation is specified, only those implementations for which simulation runs are explicitly enabled, are run.
`?impSet` *g_impSet*	The name of single implementation set or a list of implementation set names where each implementation set name must be a string. For example, `"Weekly"`.
`?runPlanRun` *g_runPlanRun*	A string that represents the name of a single *RunPlan* run name or a list of *RunPlan* run names. For example, `"Two_Stage_Opamp/OpAmp/maestro_run_plan/Run.0"`.
`?mode` *g_mode*	A string that represents the mode. The available values are: `'Local Run'`, `'Batch Run'`, `'Local with SPACE'` or `'Batch with SPACE'`. The default is `'Local Run'`.

`?waitUntilDone` *`g_waitUntilDone`*

Waits for the simulations to complete the runs before executing any other command. If this is set to `nil`, the wait is disabled. The default is `nil`.

`?rerun` *`g_rerun`*

Reruns only the unfinished or erroneous points from the loaded implementation history of the implementations. All implementations that have no incomplete or erroneous points to simulate are ignored.

The default is `nil`.

Value Returned

`t`	The simulations were successfully run or loaded.
`nil`	Failed to run simulations.

Examples

The following example shows how to use this function.

Runs all implementations.

```
sess = verifOpenCellView("test" "sample" "verifier")
verifRun(sess)
```

Runs all implementations externally sessions and waits until they have finished.

```
verifRun(sess ?mode "External Run" ?waitUntilDone t)
```

Runs a single implementation.

```
verifRun(sess ?implementations '(("test" "sample" "maestro" "Active")))
```

Runs two implementations.

```
verifRun(sess ?implementations '(("test" "sample" "maestro" "Active") ("test"
"sample2" "maestro" "Interactive.0")))
```

Reloads the results of all implementations that do not have the *Run* check box enabled.

```
verifRun(sess ?implementations verifGetImps(sess ?isRun nil))
```

Runs the specified RunPlan simulations.

```
verifRun(sessionId ?runPlanRun "Two_Stage_Opamp/OpAmp/maestro_run_plan/Active/
Nominal")
```

Runs the list of specified RunPlan simulations.

```
verifRun(sessionId ?runPlanRun list("Two_Stage_Opamp/OpAmp/maestro_run_plan/
Active/Nominal", "Two_Stage_Opamp/OpAmp/maestro_run_plan/Active/MC")
```

Runs all implementations sessions and reruns the unfinished or erroneous points.

```
verifRun(sess ?mode "Local Run" ?rerun t)
```

Related Topics

verifImpIsRun

verifSetImpRun

verifCheck

verifCheckImp

verifStop

Simulation and Results Extraction Functions

verifSetImpRun

```
verifSetImpRun(
     g_sessionId
     t_impLib
     t_impCell
     t_impView
     t_impHistory
     g_runState
     )
     => t / nil
```

Description

Sets the check state of the *Run* check box for the specified implementation in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impLib	The library name of the implementation.
t_impCell	The cell name of the implementation.
t_impView	The view name of the implementation.
t_impHistory	The history name of the implementation.
g_runState	The run state of the implementation.

Value Returned

`t`	Sets the run state of implementation successfully.
`nil`	The specified implementation does not exist or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation '`opamp090/full_diff_opamp_AC/maestro/Active`' and enables the run state for this cellview.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

```
verifSetImpRun(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active" t)
=> t
```

Related Topics

verifImpIsRun

verifRun

verifCheck

verifCheckImp

verifStop

Simulation and Results Extraction Functions

verifSetImpSetName

```
verifSetImpSetName(
     g_sessionId
     t_impSetName
     t_newImpSetName
     )
     => t / nil
```

Description

Renames the specified implementation set.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_impSetName	The name of implementation set.
t_newImpSetName	The new name of implementation set.

Value Returned

`t`	The specified implementation set is renamed successfully.
`nil`	The specified implementation set does not exist, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation set 'Weekly'. It renames this implementation set in the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifSetImpSetName(uid "Weekly" "Daily")
=> t
```

Related Topics

verifAddImpSet

verifAddImpToImpSet

verifStop

```
verifStop(
     g_sessionId
     [ ?implementation g_implementation ]
     [ ?impSet g_impSet ]
     )
     => t / nil
```

Description

Stops simulations for one or more implementations. If no implementation or implementation set is specified, then all running implementations in the session are stopped.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`?implementation` *g_implementation*	Name of single implementation or list of implementation names where each implementation name can be either a string like "Lib/Cell/View/History" or a list '("Lib" "Cell" "View" "History"). If no implementation is specified, all implementations in the session are run.
`?impSet` *g_impSet*	Name of a single implementation set or a list of implementation set names.

Value Returned

`t`	Successfully stops the simulations.
`nil`	Fails to stop simulations.

Examples

The following example starts a Verifier session with a cellview that has an implementation '`opamp090/full_diff_opamp_AC/maestro/Active`' and an implementation set 'Weekly'.

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0

verifRun(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp" "maestro_nominal"
"Active")))
Verifier: Started running implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Active' at 'Apr 15 10:00:50 2019'
verifStop(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active")))
Verifier: Stopped running implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Active' at 'Apr 15 10:00:18 2019'
=>t

verifRun(uid ?impSet "Weekly")
=>t
verifStop(uid ?impSet "Weekly")
=>t
```

Related Topics

verifImpIsRun

verifRun

verifSetImpRun

verifCheck

verifCheckImp

Simulation and Results Extraction Functions

6

Setup Library Assistant Functions

You can use several SKILL functions to work with the Setup Library Assistant (SLA) in the `nograph` or the non-GUI mode. These functions help you edit, delete, and read SLA components. The components can be corner setups, corner variables, corner model files, sweep setups, sweep variables, and verification spaces that contain sweep setups and corner setups. You can use these functions without opening a `maestro` or `verifier` view because the functions edit or read the `*sdb` files directly from the disk.

The SKILL functions that let you manage various Setup Library tasks and operations in the ADE Verifier and ADE Assembler environment can be broadly classified in the following categories.

- Functions for opening and saving setup library views.
  - slaOpenOrCreateView
  - slaSaveAndCloseView
  - slaIsViewOpened
- Functions for creating top-level components (sweep setups, corner setups, spaces, or simulation setups).
  - slaCreateCornerSetup
  - slaCreateSimulationSetup
  - slaCreateSweepSetup
  - slaCreateVerificationSpace
- Functions for adding corners, variables, model files in a corner setup or sweep setup, or modifying run options in a simulation setup.
  - slaAddCornerModelFile
  - slaAddCornerVariable
  - slaAddDocument

- ❑ slaAddSweepVariable
- ❑ slaImportCorners
- ❑ slaImportSweeps
- ❑ slaIsCornerEnabled
- ❑ slaSetCornerEnabled
- ❑ slaSetSimulationSetupRunOptionValue

■ Functions for reading setups from the setup library view.

These functions allow reading the specified setups when you open the setup library view in either edit mode or read-only mode.

- ❑ slaGetAllDocuments
- ❑ slaGetCornerModels
- ❑ slaGetCornerSetupCorners
- ❑ slaGetCornerSetups
- ❑ slaGetCornerVars
- ❑ slaGetDocumentAbsolutePath
- ❑ slaGetSimulationSetupRunOptions
- ❑ slaGetSimulationSetupRunOptionValue
- ❑ slaGetSimulationSetups
- ❑ slaGetSweepSetupVars
- ❑ slaGetSweepSetups
- ❑ slaGetVerificationSpaces

■ Functions for exporting or importing setups and verification spaces to or from the setup library view.

- ❑ slaExportSetup
- ❑ slaImportSetup

■ Functions for deleting components from the setup library view.

- ❑ slaRemoveCorner

- ❑ slaRemoveCornerModel
- ❑ slaRemoveCornerSetup
- ❑ slaRemoveCornerVariable
- ❑ slaRemoveSimulationSetup
- ❑ slaRemoveSweepSetup
- ❑ slaRemoveSweepVariable
- ❑ slaRemoveVerificationSpace

- Functions for managing parametric sets in the setup library view.
  - ❑ slaAddToParametricSet
  - ❑ slaAreParametricSetsEnabled
  - ❑ slaGetParametricSets
  - ❑ slaGroupAsParametricSet
  - ❑ slaRemoveFromParametricSet
  - ❑ slaSetParametricSetsEnabled
  - ❑ slaUngroupParametricSet
- Functions for managing reliability setups and options in the setup library view.
  - ❑ slaAddReliabilityOptions
  - ❑ slaCreateReliabilitySetup
  - ❑ slaGetReliabilityOptions
  - ❑ slaGetReliabilitySetups
  - ❑ slaIsReliabilityOptionsEnabled
  - ❑ slaRemoveReliabilityOptions
  - ❑ slaRemoveReliabilitySetup
  - ❑ slaSetReliabilityOptionsEnabled

Related Topic

Implementation Functions

Simulation and Results Extraction Functions

slaAddCornerModelFile

```
slaAddCornerModelFile(
     t_cornerSetupName
     t_cornerName
     l_list

     )
     => t / nil
```

Description

Adds a model file to a corner within a corner setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode. The function creates the specified corner if it does not exist in the corner setup.

Arguments

t_cornerSetupName	String value specifying the name of the corner setup.
t_cornerName	String value specifying the name of the corner in which the model file must be added.
l_list	List containing the name of the model file and its sections.

Value Returned

`t`	The specified model file has been added to the specified corner.
`nil`	The specified model file could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified model file to the view. The function creates a corner, `C0`, if it does not exist in `cornerSetup1`.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaAddCornerModelFile("cornerSetup1" "C0" list("./model.scs" "ff"))
=> t
```

Related Topics

slaAddCornerVariable

slaAddCornerVariable

```
slaAddCornerVariable(
     t_cornerSetupName
     t_cornerName
     t_varName
     t_varValue
     )
     => t / nil
```

Description

Adds a corner variable to a corner of a corner setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode. The function creates the specified corner if it does not exist in the corner setup.

Arguments

t_cornerSetupName	String value specifying the name of the corner setup.
t_cornerName	String value specifying the name of the corner in which the corner variable must be added.
t_varName	String value specifying the name of the variable to be added to the specified corner.
t_varValue	String value specifying the value of the variable to be added to the specified corner.

Value Returned

`t`	The specified corner variable has been added to the corner.
`nil`	The specified corner variable could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified corner variable to the view. The function creates a corner, `C0`, if it does not exist in `cornerSetup1`.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaAddCornerVariable("cornerSetup1" "C0" "vdd" "2.0:1.0:5.0")
=> t
```

Related Topics

slaAddCornerModelFile

slaAddDocument

slaAddSweepVariable

slaImportCorners

slaImportSweeps

slaSetSimulationSetupRunOptionValue

Setup Library Assistant Functions

slaAddDocument

```
slaAddDocument(
     t_docAbsoluteFilePath
     )
     => t / nil
```

Description

Adds a document to a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_docAbsoluteFilePath`	
	String value specifying the name and path of the document to be added.

Value Returned

`t`	The specified document has been added.
`nil`	The specified document could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified document to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaAddDocument("/home/project/xyz/coverage45.xls")
=> t
```

Related Topics

slaAddCornerModelFile

slaAddCornerVariable

slaAddSweepVariable

slaImportCorners

slaAddReliabilityOptions

```
slaAddReliabilityOptions(
     t_setupName
     t_optionsName
     )
     => t / nil
```

Description

Adds new reliability options in the Setup Library Assistant with default values.

Arguments

`t_setupName`	Specifies an existing reliability setup name.
`t_optionsName`	Specifies the name of the reliability options.

Value Returned

`t`	The reliability options are added successfully.
`nil`	Indicates that the reliability setup name or the options are invalid.

Examples

Opens a setup library view and adds new reliability options in Setup Library assistant.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaAddReliabilityOptions("ReliabilitySetup1" "Reliability1")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaCreateReliabilitySetup

slaGetReliabilityOptions

slaGetReliabilitySetups

slaAddSweepVariable

```
slaAddSweepVariable(
     t_sweepSetupName
     t_varName
     t_varValue
     )
     => t / nil
```

Description

Adds a sweep variable to a sweep setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_sweepSetupName	String value specifying the name of the sweep setup.
t_varName	String value specifying the name of the variable to be added to the sweep setup.
t_varValue	String value specifying the value of the variable to be added to the sweep setup.

Value Returned

`t`	The specified sweep variable was added to the specified setup.
`nil`	The specified sweep variable could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified sweep variable to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaAddSweepVariable("sweepSetup1" "vdd" "2.0:1.0:5.0")
=> t
```

Related Topics

slaAddCornerModelFile

slaAddCornerVariable

slaAddToParametricSet

```
slaAddToParametricSet(
     t_setupName
     g_setId
     t_varName
     )
     => t / nil
```

Description

Adds a variable to the specified parametric set in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_setupName`	Specifies the name of the sweep setup.
`g_setId`	Specifies an integer or string number to denote the parametric set ID. For example, `1` or "`1`".
`t_varName`	Specifies the name of the variable to be added to the specified parametric set.

Value Returned

`t`	The variable is added to the specified parametric set.
`nil`	The simulation setup, parametric set, or variable with the specified name does not exist, or the command failed.

Examples

Opens a setup library view in edit mode and adds a variable to a parametric set.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
```

Invalid arguments are specified while adding a variable to the parametric set.

```
slaAddToParametricSet(SweepSetup1 "XXX" "XXX")
ERROR (ADE-11414): Cannot run the function because the parametric set 'XXX' is
invalid. Specify a valid parametric set name and rerun the function.
=> nil
```

The variable to be added to the parametric set is invalid.

```
slaAddToParametricSet(SweepSetup1 1 "XXX")
ERROR (ADE-11414): Cannot run the function because the variable 'XXX' is invalid.
Specify a valid variable name and rerun the function.
=> nil
```

Valid arguments are specified while adding a variable to the parametric set.

```
slaAddToParametricSet(SweepSetup1 1 "var1")
=> t
```

Related Topics

slaAreParametricSetsEnabled

slaGetParametricSets

slaGroupAsParametricSet

slaRemoveFromParametricSet

slaSetParametricSetsEnabled

slaUngroupParametricSet

Setup Library Assistant Functions

slaAreParametricSetsEnabled

```
slaAreParametricSetsEnabled(
     t_setupName
     )
     => t / nil
```

Description

Checks if all parametric sets are enabled for the specified sweep setup in the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

t_setupName	Specifies the name of the sweep setup.

Value Returned

`t`	All parametric sets are enabled and displayed in the Setup Library Assistant.
`nil`	All parametric sets are disabled and hidden in the Setup Library Assistant.

Examples

Opens a setup library view in edit mode and checks if all parametric sets are enabled or disabled.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
slaAreParametricSetsEnabled("SweepSetup1")
=> t
```

Related Topics

slaAddToParametricSet

slaGetParametricSets

slaGroupAsParametricSet

slaRemoveFromParametricSet

slaCreateCornerSetup

```
slaCreateCornerSetup(
     t_setupName
     )
     => t / nil
```

Description

Creates a corner setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_setupName	String value specifying the name of the corner setup to be created and added to the view.

Value Returned

`t`	The specified corner setup has been created.
`nil`	The specified corner setup could not be created.

Examples

Opens a setup library view in the edit mode and adds the specified corner setup to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaCreateCornerSetup("cornerSetup1")
=> t
```

Related Topics

slaCreateSimulationSetup

slaCreateSweepSetup

slaCreateVerificationSpace

Setup Library Assistant Functions

slaCreateReliabilitySetup

```
slaCreateReliabilitySetup(
     t_setupName
     )
     => t / nil
```

Description

Creates a new reliability setup in the Setup Library assistant with the specified name.

Arguments

t_setupName	Specifies a name to be assigned to the new reliability setup.

Value Returned

`t`	The reliability setup is added to the Setup Library assistant.
`nil`	The reliability setup is not created because the setup name is invalid.

Examples

Opens a setup library view and creates a new reliability setup.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaCreateReliabilitySetup("ReliabilitySetup1")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaGetReliabilityOptions

slaGetReliabilitySetups

slaIsReliabilityOptionsEnabled

slaRemoveReliabilityOptions

slaCreateSimulationSetup

```
slaCreateSimulationSetup(
     t_setupName
     )
     => t / nil
```

Description

Creates a simulation setup in the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

t_setupName	Specifies the name of the simulation setup to be created and added to the view.

Value Returned

`t`	The specified simulation setup has been created.
`nil`	The specified simulation setup could not be created.

Examples

Opens a setup library view and adds the specified simulation setup to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaCreateSimulationSetup("SimpleMC_10Points")
=> t
```

Related Topics

slaCreateCornerSetup

slaCreateSweepSetup

slaCreateVerificationSpace

Setup Library Assistant Functions

slaCreateSweepSetup

```
slaCreateSweepSetup(
     t_setupName
     )
     => t / nil
```

Description

Creates a sweep setup in a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_setupName`	String value specifying the name of the corner setup to be added to the view.

Value Returned

`t`	The specified sweep setup has been added.
`nil`	The specified sweep setup could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified sweep setup to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaCreateSweepSetup("sweepSetup1")
=> t
```

Related Topics

slaCreateCornerSetup

slaCreateSimulationSetup

slaCreateVerificationSpace

Setup Library Assistant Functions

slaCreateVerificationSpace

```
slaCreateVerificationSpace(
     t_spaceName
     t_sweepSetupName
     t_cornerSetupName
     [ t_simulationSetupName ]
     )
     => t / nil
```

Description

Creates a verification space in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_spaceName	Specifies the name of the verification space to be added.
t_sweepSetupName	Specifies the name of the sweep setup to which the verification space is to be added.
t_cornerSetupName	Specifies the name of the corner setup to which the verification space is to be added.
t_simulationSetupName	Specifies the name of the simulation setup to which the verification space is to be added.

Value Returned

`t`	The specified verification space has been added.
`nil`	The specified verification space could not be added.

Examples

Opens a setup library view in edit mode.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
```

Creates a verification space `Space1` using existing setups.

```
slaCreateVerificationSpace("Space1" "SweepSetup1" "CornerSetup1" "Monte Carlo
Setup1")
=> t
```

Returns the names of existing verification spaces with the setups they contain.

```
slaGetVerificationSpaces()
=> (("Space1" ( "SweepSetup1" "CornerSetup1" "Monte Carlo Setup1")))
```

Related Topics

slaCreateCornerSetup

slaCreateSimulationSetup

slaCreateSweepSetup

slaRemoveSimulationSetup

Setup Library Assistant Functions

slaExportSetup

```
slaExportSetup(
     t_fileName
     )
     => t / nil
```

Description

Exports the sweep setup, corner setup, simulation setup, and verification spaces to the SDB file in the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

`t_fileName`	Specifies the name of the `SDB` file.

Value Returned

`t`	The setups have been exported to the specified file.
`nil`	The setups could not be exported to the specified file.

Examples

Opens a setup library view and exports the setups to the SDB file.

```
slaOpenOrCreateView("Two_Stage_Opamp" "ProjectSetup" "setupLib")
=> t
slaExportSetup("./setup.sdb")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaImportSetup

Setup Library Assistant Functions

slaGetAllDocuments

```
slaGetAllDocuments(
     )
     => l_listOfDocuments
```

Description

Retrieves a list of documents from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

None

Value Returned

`l_listOfDocuments`	A list of documents saved in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves a list of documents from this view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetAllDocuments()
=> ("coverage45nm.xls")
```

Related Topics

slaGetDocumentAbsolutePath

Setup Library Assistant Functions

slaGetCornerModels

```
slaGetCornerModels(
     t_cornerSetupName
     t_cornerName
     )
     => l_listOfCornerModels
```

Description

Retrieves a list of corner models of corner setup from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup.
`t_cornerName`	String value specifying the name of the corner that contains the corner models.

Value Returned

`l_listOfCornerModels`	A list of all corner models saved in the specified corner.

Examples

Opens a setup library view in the edit mode and retrieves the specified list of corner models from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetCornerModels("cornerSetup1" "C0")
=> (("gpdk090.scs" "path/to/file/gpdk090.scs" "\"SF\" \"FS\""))
```

Related Topics

slaGetCornerSetupCorners

slaGetCornerSetups

slaGetCornerVars

Setup Library Assistant Functions

slaGetCornerSetupCorners

```
slaGetCornerSetupCorners(
     t_cornerSetupName
     )
     => l_listOfCorners
```

Description

Retrieves a list of corners for a specified corner setup from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup.

Value Returned

`l_listOfCorners`	A list of corners saved in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves the specified list of corners from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetCornerSetupCorners("cornerSetup1")
=> ("C0" "C1")
```

Related Topics

slaGetCornerModels

slaGetCornerSetups

slaGetCornerVars

Setup Library Assistant Functions

slaGetCornerSetups

```
slaGetCornerSetups(
     )
     => l_listOfCornerSetups
```

Description

Retrieves a list of corner setups from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

None

Value Returned

`l_listOfCornerSetups`	
	A list of corner setups saved in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves the specified corner setups from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetCornerSetups()
=> ("cornerSetup1" "cornerSetup2")
```

Related Topics

slaGetAllDocuments

slaGetCornerModels

slaGetCornerSetupCorners

slaGetCornerVars

Setup Library Assistant Functions

slaGetCornerVars

```
slaGetCornerVars(
     t_cornerSetupName
     t_cornerName
     )
     => l_listOfCorners
```

Description

Retrieves a list of corner variables of the specified corner setup from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_cornerSetupName	String value specifying the name of the corner setup within the setup library view.
t_cornerName	String value specifying the name of the corner that contains the corner variables.

Value Returned

l_listOfCornerVariables	A list of all corner variables saved in the specified corner of the corner setup.

Examples

Opens a setup library view in the edit mode and retrieves the specified corner variables from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetCornerVars("cornerSetup1" "C0")
=> (("gain" "11:1:12"))
```

Related Topics

slaGetAllDocuments

slaGetCornerModels

slaGetCornerSetupCorners

slaGetCornerSetups

Setup Library Assistant Functions

slaGetDocumentAbsolutePath

```
slaGetDocumentAbsolutePath(
     t_docName
     )
     => t_docAbsoluteFilePath
```

Description

Retrieves the absolute path of a document in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_docName`	String value specifying the name of the document.

Value Returned

`t_docAbsoluteFilePath`	Absolute path of the document in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves the absolute path of a specified document.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetDocumentAbsolutePath("coverage45.xls")
=> "<setupLibrary_sdb_directory_path>/documents/coverage45.xls"
```

Related Topics

slaGetAllDocuments

Setup Library Assistant Functions

slaGetParametricSets

```
slaGetParametricSets(
     t_setupName
     [ ?getVariables g_getVariables ]
     )
     => l_psets / nil
```

Description

Retrieves a list of parametric sets from the specified sweep setup in the setup library cellview that is opened using the `slaOpenOrCreateView` function.

Arguments

`t_setupName`	Specifies the name of the sweep setup.
`?getVariables` *`g_getVariables`*	Returns a list of pairs, where each pair is the name of the parametric set and its variables.

Value Returned

`l_psets`	Returns a list of parametric sets from the sweep setup.
`nil`	Either the specified setup does not exist or it contains no parametric sets.

Examples

Opens a setup library view in edit mode and adds a variable to a parametric set.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
```

Invalid arguments are specified while retrieving parametric sets from a sweep setup.

```
slaGetParametricSets("XXX")
ERROR (ADE-11414): Cannot run the function because the sweep setup 'XXX' is invalid.
Specify a valid sweep setup name and rerun the function.
=> nil
```

Valid arguments are specified while retrieving parametric sets from a sweep setup.

```
slaGetParametricSets(SweepSetup1)
=> ("1" "2" "3")
```

```
slaGetParametricSets(SweepSetup1 ?getVariables t)
=> (("1" ("var4" "var5")) ("2" ("var7" "var1" "var3")) ("3" ("var9" "var8")))
```

Related Topics

slaAddToParametricSet

slaAreParametricSetsEnabled

slaGroupAsParametricSet

slaRemoveFromParametricSet

slaSetParametricSetsEnabled

slaUngroupParametricSet

Setup Library Assistant Functions

slaGetReliabilityOptions

```
slaGetReliabilityOptions(
     t_setupName
     )
     => l_optionsNames
```

Description

Retrieves a list of names of options set for a reliability setup.

Arguments

t_setupName	Specifies the name of an existing reliability setup.

Value Returned

l_optionsNames	List of reliability option names included in the specified reliability setup.

Examples

Opens a setup library view and creates the realibility setup and options in the setup.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaCreateReliabilitySetup("ReliabilitySetup1")
=> t
```

Adds the reliability options in the setup.

```
slaAddReliabilityOptions("ReliabilitySetup1" "Reliability1")
=> t
slaAddReliabilityOptions("ReliabilitySetup1" "Reliability2")
=> t
slaAddReliabilityOptions("ReliabilitySetup1" "Reliability3")
=> t
```

Gets the name list of the reliability setup.

```
slaGetReliabilityOptions("ReliabilitySetup1")
=> ("Reliability1" "Reliability2" "Reliability3")
```

Saves and closes the setup library cellview.

```
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaCreateReliabilitySetup

slaGetReliabilitySetups

slaIsReliabilityOptionsEnabled

slaRemoveReliabilityOptions

slaRemoveReliabilitySetup

slaSetReliabilityOptionsEnabled

slaGetReliabilitySetups

```
slaGetReliabilitySetups(
     )
     => l_setupsNames
```

Description

Retrieves a list of all reliability setups that are included in the setup library cellview.

Arguments

None

Value Returned

l_setupsNames	List of all of reliability setups included in the opened setup library cellview.

Examples

Opens a setup library view, creates reliability options in the reliability setup, and then retrieves these options.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaCreateReliabilitySetup("ReliabilitySetup1")
=> t
slaCreateReliabilitySetup("ReliabilitySetup2")
=> t
slaCreateReliabilitySetup("ReliabilitySetup3")
=> t
slaGetReliabilitySetups()
=> ("ReliabilitySetup1" "ReliabilitySetup2" "ReliabilitySetup3")
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaCreateReliabilitySetup

slaGetReliabilityOptions

slaGetSimulationSetupRunOptions

```
slaGetSimulationSetupRunOptions(
     t_setupName
     )
     => l_runOptions / nil
```

Description

Retrieves the list of run options for the specified simulation setup of the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

t_setupName	Specifies the name of the simulation setup.

Value Returned

l_runOptions	List of the run options for the specified simulation setup.
nil	The simulation setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view and retrieves the run options for the specified simulation setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetSimulationSetupRunOptions("SimpleMC_10Points")
=> ("mcnumpoints" "mcmethod" "samplingmode" "montecarloseed")
```

Returns an error message if the specified simulation setup does not exist.

```
slaGetSimulationSetupRunOptions("XXX")
ERROR (ADE-11414): Cannot run the function because the simulation setup 'XXX' is
invalid. Specify a valid simulation setup name and rerun the function.
=> nil
```

Related Topics

slaGetSimulationSetupRunOptionValue

slaGetSimulationSetups

Setup Library Assistant Functions

slaGetSimulationSetupRunOptionValue

```
slaGetSimulationSetupRunOptionValue(
     t_setupName
     t_optionName
     )
     => t_optionValue / nil
```

Description

Retrieves the value of the run option for the specified simulation setup of the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

t_setupName	Specifies the name of the simulation setup.
t_optionName	Specifies the name of the run option whose value is required.

Value Returned

t_optionValue	Value of the run option for the specified simulation setup.
nil	The simulation setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view and retrieves the value of the run option for the specified simulation setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcnumpoints")
=> "10"
```

Returns an error message if the specified simulation setup does not exist.

```
slaGetSimulationSetupRunOptionValue("XXX" "mcnumpoints")
ERROR (ADE-11414): Cannot run the function because the simulation setup 'XXX' is
invalid. Specify a valid simulation setup name and rerun the function.
=> nil
```

Related Topics

sláGetSimulationSetupRunOptions

slaGetSimulationSetups

Setup Library Assistant Functions

slaGetSimulationSetups

```
slaGetSimulationSetups(
    )
    => l_simulationSetups / nil
```

Description

Retrieves the list of simulation setups from the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

None

Value Returned

`l_simulationSetups`	List of the simulation setups in the opened setup library view.
`nil`	No simulation setups exist in the opened setup library view, or the command failed.

Examples

Opens a setup library view and retrieves the list of simulation setups from the opened setup library view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetSimulationSetups()
=> ("Monte Carlo Setup1" "Monte Carlo Setup1")
```

Related Topics

slaGetSimulationSetupRunOptions

slaGetSimulationSetupRunOptionValue

Setup Library Assistant Functions

slaGetSweepSetups

```
slaGetSweepSetups(
     )
     => l_listOfSweepSetups
```

Description

Retrieves a list of sweep setups from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

None

Value Returned

`l_listOfSweepSetups`	A list of sweep setups saved in the setup library view.

Examples

The following example retrieves the specified sweep setups from the setup library view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetSweepSetups()
=> ("sweepSetup1"
    "sweepSetup2"
   )
```

Related Topics

slaGetSweepSetupVars

slaGetVerificationSpaces

slaRemoveSweepSetup

slaRemoveSweepVariable

Setup Library Assistant Functions

slaGetSweepSetupVars

```
slaGetSweepSetupVars(
     t_sweepSetupName
     )
     => l_listOfSweepVars
```

Description

Retrieves a list of sweep setup variables from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_sweepSetupName	String value specifying the name of the sweep setup.

Value Returned

l_listOfSweepVars	A list of all sweep variables saved in the specified sweep setup.

Examples

Opens a setup library view in the edit mode and retrieves a list of sweep variables from the specified view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetSweepSetupVars("sweepSetup1")
=> (("gain" "15")
    ("var1" "455")
   )
```

Related Topics

slaGetSweepSetups

slaGetVerificationSpaces

slaRemoveSweepSetup

slaRemoveSweepVariable

Setup Library Assistant Functions

slaGetVerificationSpaces

```
slaGetVerificationSpaces(
     )
     => l_listOfLists
```

Description

Retrieves a list of verification spaces from the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

None

Value Returned

`l_listOfLists`	A list of verification spaces saved in the setup library view.

Examples

The following example returns the names of existing verification spaces with the setups that they contain.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaGetVerificationSpaces()
=> (("Space1" ( "SweepSetup1" "CornerSetup1" "Monte Carlo Setup1")))
```

Related Topics

slaGetCornerModels

slaGetCornerSetupCorners

slaGetCornerSetups

slaGetSimulationSetups

slaGetVerificationSpaces

Setup Library Assistant Functions

slaGroupAsParametricSet

```
slaGroupAsParametricSet(
     g_setupName
     l_varNames
     )
     => t_setName / nil
```

Description

Groups the variables as parametric set in sweep setup in the setup library view that is opened using the `slaOpenOrCreateView` function in editable mode.

Arguments

g_setupName	Specifies the name of the sweep setup.
l_varNames	List of variables to be grouped as a parametric set.

Value Returned

t_setName	Name of the parametric set.
nil	The sweep setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view in editable mode and groups the variables as a parametric set.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
```

Returns `nil` when invalid arguments are specified while grouping variables as a parametric set.

```
slaGroupAsParametricSet("SweepSetup1" list("var1"))
WARNING (ADE-2342): At least two variables or parameters should be selected to make
a parametric set.
=> nil
slaGroupAsParametricSet("SweepSetup1" list("XXX1" "var1" "var2" "XXX2"))
ERROR (ADE-11414): Cannot run the function because the variable 'XXX1, XXX2' is
invalid. Specify a valid variable name and rerun the function.
=> nil
```

Returns the name or ID of the parametric set when a valid argument is specified while grouping variables as a parametric set.

```
slaGroupAsParametricSet("SweepSetup1" list("var1" "var2"))
=> "0"
slaGroupAsParametricSet("SweepSetup1" list("var3" "var4" "var5"))
=> "1"
```

Related Topics

slaAddToParametricSet

slaAreParametricSetsEnabled

slaGetParametricSets

slaRemoveFromParametricSet

slaSetParametricSetsEnabled

slaUngroupParametricSet

Setup Library Assistant Functions

slaImportCorners

```
slaImportCorners(
     t_cornerSetupName
     t_absoluteFilePath
     )
     => t / nil
```

Description

Imports corners into the setup library corner setup opened using the `slaOpenOrCreateView` function in edit mode. The file must be in CSV or SDB format.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup within the setup library view.
`t_absoluteFilePath`	String value specifying the absolute path of the file that contains the corners.

Value Returned

`t`	The specified corners were imported into the specified setup.
`nil`	The specified corners could not be imported.

Examples

Opens a setup library view in the edit mode and imports corners from the specified file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaImportCorners("cornerSetup1" "path/to/file/corner.csv")
=> t
```

Related Topics

slaAddCornerModelFile

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaImportSweeps

slaSetSimulationSetupRunOptionValue

Setup Library Assistant Functions

slaImportSetup

```
slaImportSetup(
     t_fileName
     )
     => t / nil
```

Description

Imports the sweep setup, corner setup, simulation setup, and verification spaces from the `SDB` file in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_fileName`	Specifies the name of the `SDB` file.

Value Returned

`t`	The setups have been imported from the specified file.
`nil`	The SDB file is invalid or does not exist, or the command failed.

Examples

Opens a setup library view and imports the setups from the SDB file.

```
slaOpenOrCreateView("Two_Stage_Opamp" "ProjectSetup" "setupLib")
=> t
slaImportSetup("./setup.sdb")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaExportSetup

Setup Library Assistant Functions

slaImportSweeps

```
slaImportSweeps(
     t_sweepSetupName
     t_absoluteFilePath
     )
     => t / nil
```

Description

Imports sweeps into the corner setup in a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode. The supported file extensions are `csv` and `sdb`.

Arguments

`t_sweepSetupName`	String value specifying the name of the sweep setup within the setup library view.
`t_absoluteFilePath`	String value specifying the absolute path of the file that contains the sweeps.

Value Returned

`t`	The specified sweeps were imported into the specified setup.
`nil`	The specified sweeps could not be imported.

Examples

Opens a setup library view in the edit mode and imports sweeps from the specified file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaImportSweeps("sweepSetup1" "path/to/file/sweep.csv")
=> t
```

Related Topics

slaAddCornerModelFile

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaImportCorners

slaSetSimulationSetupRunOptionValue

Setup Library Assistant Functions

slaIsCornerEnabled

```
slaIsCornerEnabled(
     t_cornerSetupName
     t_cornerName
     )
     => t / nil
```

Description

Retrieves the status of the specified corner within a corner setup in the a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode. The function displays a warning if an invalid corner setup name or corner name is specified.

Arguments

`t_cornerSetupName`	The name of the corner setup.
`t_cornerName`	The name of the corner.

Value Returned

`t`	The specified corner is enabled.
`nil`	The specified corner is disabled.

Examples

The following example opens a setup library view in edit mode and retrieves the enabled or disabled status of the specified corner.

```
slaOpenOrCreateView("Two_Stage_Opamp" "project45nm" "setupLib")
=> t
slaIsCornerEnabled("CornerSetup1" "C0")
=>t
```

Related Topics

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaImportCorners

slaIsReliabilityOptionsEnabled

```
slaIsReliabilityOptionsEnabled(
     t_setupName
     t_optionsName
     )
     => t / nil
```

Description

Retrieves the enabled or disabled status of the reliability options included in a reliability setup in the setup library view that is opened using slaOpenOrCreateView in the edit mode. The function reports a warning if you specify an incorrect name for a reliability setup or option.

Arguments

t_setupName	Specifies the name of the reliability setup.
t_optionsName	Specifies the name of the reliability options.

Value Returned

`t`	The reliability options are enabled.
`nil`	The reliability options are disabled.

Examples

Opens a setup library view and retrieves the enabled or disabled status of the reliability options.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaIsReliabilityOptionsEnabled("ReliabilitySetup1" "Reliability1")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaCreateReliabilitySetup

slaIsViewOpened

```
slaIsViewOpened(
     )
     => t / nil
```

Description

Checks whether a setup library view is opened or closed.

Arguments

None.

Value Returned

`t`	The setup library view is opened.
`nil`	The setup library view is closed.

Examples

The following example opens a setup library view in edit mode and checks if the setup library view is opened or closed.

```
slaIsViewOpened()
=> nil
slaOpenOrCreateView("amsPLL" "TOP_verification_Space" "setupLib")
=> t
slaIsViewOpened()
=> t
slaSaveAndCloseView()
=> t
slaIsViewOpened()
=> nil
```

Related Topics

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaSetCornerEnabled

```
slaSetCornerEnabled(
     t_cornerSetupName
     t_cornerName
     g_enabled
     )
     => t / nil
```

Description

Sets the enabled or disabled status of the specified corner within a corner setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode. The function displays a warning if an invalid corner setup name or corner name is specified.

Arguments

t_cornerSetupName	The name of the corner setup.
t_cornerName	The name of the corner.
g_enabled	Specifies whether the corner for the specified sweep setup should be enabled (`t`) or disabled (`nil`).

Value Returned

`t`	The specified corner is set to the user-specified enabled or disabled state.
`nil`	The corner setup with the specified name does not exist or the command is unsuccessful.

Examples

The following example opens a setup library view in edit mode and retrieves the enabled or disabled status of the specified corner.

```
slaOpenOrCreateView("Two_Stage_Opamp" "project45nm" "setupLib")
=> t
slaSetCornerEnabled("CornerSetup1" "C0" t)
=> t
```

Related Topics

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaOpenOrCreateView

```
slaOpenOrCreateView(
     t_libName
     t_cellName
     t_viewName
     [ g_readOnly ]
     )
     => t / nil
```

Description

When this function is run in edit mode, it opens the specified view in edit mode if the view exists. Otherwise, it creates a new view. In read-only mode, the function opens the provided view in read-only mode. In both cases, invoking this function checks out the ADE Verifier license. You can initialize the view from the Virtuoso CIW or by using the `*il` script.

Arguments

t_libName	String value specifying the library name.
t_cellName	String value specifying the cell name.
t_viewName	String value specifying the view name.
g_readOnly	Boolean value that specifies if the view must be opened in read-only mode or edit mode. Setting the value to `t` opens the cellview in read-only mode. The default is `nil`.

Value Returned

`t`	The specified view has been opened.
`nil`	The specified view could not be opened.

Examples

Opens a setup library view in the edit mode and checks out the Verifier license. It creates the setup library view if the given view does not exist.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
```

Opens the setup library view in read-only mode and checks out the Verifier license.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib" t)
=> t
```

Related Topics

slaSaveAndCloseView

Setup Library Assistant Functions

slaRemoveCorner

```
slaRemoveCorner(
     t_cornerSetupName
     t_cornerName
     )
     => t / nil
```

Description

Removes a corner from the corner setup in a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_cornerSetupName	String value specifying the name of the corner setup within the setup library view.
t_cornerName	String value specifying the name of the corner to be removed.

Value Returned

`t`	The specified corner has been removed.
`nil`	The specified corner could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveCorner("cornerSetup1" "C0")
=> t
```

Related Topics

slaRemoveCornerSetup

slaRemoveSimulationSetup

slaRemoveSweepSetup

slaRemoveVerificationSpace

Setup Library Assistant Functions

slaRemoveCornerModel

```
slaRemoveCornerModel(
     t_cornerSetupName
     t_cornerName
     t_modelName
     )
     => t / nil
```

Description

Removes a corner model file from the corner setup in a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup within the setup library view.
`t_cornerName`	String value specifying the name of the corner that contains the model file to be removed.
`t_modelName`	String value specifying the name of the model file to be removed.

Value Returned

`t`	The specified corner model file has been removed.
`nil`	The specified corner model file could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner model file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveCornerModel("cornerSetup1" "C0" "model.scs")
=> t
```

Related Topics

slaRemoveCornerSetup

slaRemoveCornerSetup

```
slaRemoveCornerSetup(
     t_cornerSetupName
     )
     => t / nil
```

Description

Removes a corner setup from a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup.

Value Returned

`t`	The specified corner setup has been removed.
`nil`	The specified corner setup could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveCornerSetup("cornerSetup1")
=> t
```

Related Topics

slaRemoveCorner

slaRemoveSimulationSetup

slaRemoveSweepSetup

slaRemoveVerificationSpace

Setup Library Assistant Functions

slaRemoveCornerVariable

```
slaRemoveCornerVariable(
     t_cornerSetupName
     t_cornerName
     t_varName
     )
     => t / nil
```

Description

Removes a corner variable from corner setup of a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_cornerSetupName`	String value specifying the name of the corner setup.
`t_cornerName`	String value specifying the name of the corner from which the corner variable must be removed.
`t_varName`	String value specifying the name of the corner variable to be removed from the specified corner.

Value Returned

`t`	The specified corner variable was removed.
`nil`	The specified corner variable could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner variable.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveCornerVariable("cornerSetup1" "C0" "vdd")
=> t
```

Related Topics

slaRemoveCorner

slaRemoveSweepVariable

Setup Library Assistant Functions

slaRemoveFromParametricSet

```
slaRemoveFromParametricSet(
     t_setupName
     g_setId
     t_varName
     )
     => t / nil
```

Description

Removes a variable from the specified parametric set in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_setupName	The name of the sweep setup.
g_setId	An integer or string number to denote the parametric set ID. For example, 1 or "1".
t_varName	The name of the variable to be removed from the specified parametric set.

Value Returned

`t`	The variable is removed from the specified parametric set.
`nil`	The sweep setup, parametric set, or variable with the specified name does not exist, or the command failed.

Examples

Opens a setup library view in edit mode.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
```

Invalid arguments are specified while removing a variable from the parametric set.

```
slaRemoveFromParametricSet("SweepSetup1" "XXX" "XXX")
=> ERROR (ADE-11414): Cannot run the function because the parametric set 'XXX' is
invalid. Specify a valid parametric set name and rerun the function.
=> nil
```

```
slaRemoveFromParametricSet("SweepSetup1" 1 "XXX")
=> ERROR (ADE-11414): Cannot run the function because the variable 'XXX' is invalid.
```

```
Specify a valid variable name and rerun the function.
=> nil
```

Valid arguments are specified while removing a variable from the parametric set.

```
slaRemoveFromParametricSet("SweepSetup1" 1 "var1")
=> t
```

Related Topics

slaAddToParametricSet

slaAreParametricSetsEnabled

slaGetParametricSets

slaGroupAsParametricSet

slaSetParametricSetsEnabled

slaUngroupParametricSet

Setup Library Assistant Functions

slaRemoveReliabilityOptions

```
slaRemoveReliabilityOptions(
     t_setupName
     t_optionName
     )
     => t / nil
```

Description

Removes the specified reliability options from a given reliability setup.

Arguments

t_setupName	Specifies the name of the reliability setup.
t_optionName	Specifies the name of the reliability option.

Value Returned

`t`	The reliability options are removed successfully.
`nil`	The name of the reliability setup or the option is invalid.

Examples

Opens a setup library view and removes the given reliability option from the specified reliability setup.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaRemoveReliabilityOptions("ReliabilitySetup1" "Reliability1")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaCreateReliabilitySetup

slaGetReliabilityOptions

slaRemoveReliabilitySetup

```
slaRemoveReliabilitySetup(
     t_setupName
     )
     => t / nil
```

Description

Removes the specified reliability setup from a setup library cellview.

Arguments

t_setupName	Specifies the name of the reliability setup to be removed.

Value Returned

t	The reliability setup is removed from the setup library cellview.
nil	The reliability setup does not exist.

Examples

Opens a setup library view and removes the specified reliability setup.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaRemoveReliabilitySetup("ReliabilitySetup1")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaCreateReliabilitySetup

slaGetReliabilityOptions

slaGetReliabilitySetups

slaIsReliabilityOptionsEnabled

slaRemoveSimulationSetup

```
slaRemoveSimulationSetup(
     t_setupName
     )
     => t / nil
```

Description

Removes a simulation setup from the setup library view that is opened using the `slaOpenOrCreateView` function.

Using this function also clears the simulation setup fields for the corresponding verification space. If the verification space contains only a simulation setup field, the verification space is removed along with the simulation setup.

Arguments

t_setupName	Specifies the name of the simulation setup.

Value Returned

`t`	The specified simulation setup was removed.
`nil`	The simulation setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view and removes the specified simulation setup from the opened setup library view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveSimulationSetup("SimpleMC_10Points")
=> t
```

Returns an error message if the specified simulation setup does not exist.

```
slaRemoveSimulationSetup("XXX")
ERROR (ADE-11414): Cannot run the function because the simulation setup 'XXX' is
invalid. Specify a valid simulation setup name and rerun the function.
=> nil
```

Related Topics

slaCreateSimulationSetup

slaCreateVerificationSpace

slaRemoveCorner

slaRemoveCornerSetup

slaRemoveSweepSetup

slaRemoveVerificationSpace

Setup Library Assistant Functions

slaRemoveSweepSetup

```
slaRemoveSweepSetup(
     t_sweepSetupName
     )
     => t / nil
```

Description

Removes a sweep setup from a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_sweepSetupName`	String value specifying the name of the sweep setup within the setup library view.

Value Returned

`t`	The specified sweep setup was removed from the specified setup.
`nil`	The specified sweep setup could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified sweep setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveSweepSetup("sweepSetup1")
=> t
```

Related Topics

slaRemoveCornerSetup

slaRemoveSimulationSetup

slaRemoveSweepSetup

slaRemoveSweepVariable

Setup Library Assistant Functions

slaRemoveSweepVariable

```
slaRemoveSweepVariable(
     t_sweepSetupName
     t_varName
     )
     => t / nil
```

Description

Removes a specified sweep variable from the corner setup of a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_sweepSetupName`	String value specifying the name of the sweep setup.
`t_varName`	String value specifying the name of the sweep variable to be removed from the sweep setup.

Value Returned

`t`	The specified sweep variable has been removed.
`nil`	The specified sweep variable could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified sweep variable.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveSweepVariable("sweepSetup1" "vdd")
=> t
```

Related Topics

slaRemoveCornerVariable

slaRemoveSimulationSetup

slaRemoveSweepSetup

slaRemoveVerificationSpace

Setup Library Assistant Functions

slaRemoveVerificationSpace

```
slaRemoveVerificationSpace(
     t_spaceName
     )
     => t / nil
```

Description

Removes a verification space from a setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

`t_spaceName`	String value specifying the name of the verification space.

Value Returned

`t`	The specified verification space was removed.
`nil`	The specified verification space could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified verification space.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaRemoveVerificationSpace("space1")
=> t
```

Related Topics

slaRemoveCornerSetup

slaRemoveSimulationSetup

slaRemoveSweepSetup

slaRemoveVerificationSpace

Setup Library Assistant Functions

slaSaveAndCloseView

```
slaSaveAndCloseView(
     [ ?overrideDM g_overrideDM ]
     [ ?autoCheckInNewView g_autoCheckInNewView ]
     [ ?checkInComment g_checkInComment ]
     )
     => t / nil
```

Description

Saves the specified view after editing. If a setup library view has been opened with the `slaOpenOrCreateView` function, the `slaSaveAndCloseView` function must be invoked after adding or removing components in the setup library assistant to save all the changes in the `setupdb`. Invoking the `slaSaveAndCloseView()` function releases or checks in the ADE Verifier license.

Arguments

`?overrideDM` *`g_overrideDM`*	
	Boolean value that checks if the view can be checked in automatically in a DM setup. When set to the default value of `nil`, it checks in the view if the DM allows it. In such a case, the view is checked in automatically without a prompt. Otherwise, the view is not checked in and remains unmanaged. The DM settings are overridden when set to `t`.
`?autoCheckInNewView` *`g_autoCheckInNewView`*	
	Boolean value that when set to `t`, ignores the DM settings and checks in the view with or without comments. The default is `nil`.
`?checkInComment` *`g_checkInComment`*	
	String value that specifies any comments.

Value Returned

`t`	The specified view was saved.
`nil`	The specified view could not be saved.

Examples

The following example saves and closes the open setup library view and checks in or releases the Verifier license.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaOpenOrCreateView

Setup Library Assistant Functions

slaSetParametricSetsEnabled

```
slaSetParametricSetsEnabled(
     t_setupName
     g_enabled
     )
     => t / nil
```

Description

Enables or disables all parametric sets for the specified sweep setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

t_setupName	The name of the sweep setup.
g_enabled	Specifies whether the parametric sets for the specified sweep setup should be enabled (`t`) or disabled (`nil`).

Value Returned

`t`	All parametric sets are enabled or disabled.
`nil`	The sweep setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view in editable mode and disables all parametric sets.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t

slaAreParametricSetsEnabled("SweepSetup1")
=> t

slaSetParametricSetsEnabled("SweepSetup1" nil)
=> t
slaAreParametricSetsEnabled("SweepSetup1")
=> nil
```

Related Topics

slaAddToParametricSet

slaAreParametricSetsEnabled

slaGetParametricSets

slaGroupAsParametricSet

slaRemoveFromParametricSet

slaUngroupParametricSet

Setup Library Assistant Functions

slaSetReliabilityOptionsEnabled

```
slaSetReliabilityOptionsEnabled(
     t_setupname
     t_optionName
     g_enabled
     )
     => t / nil
```

Description

Enables or disables the specified reliability option in a reliability setup.

Arguments

`t_setupName`	Specifies the name of the reliability setup.
`t_optionName`	Specifies the name of the reliability option.
`g_enabled`	Specifies whether the reliability option included in the specified reliability setup must be enabled or disabled. Possible values are `t` and `nil`.

Value Returned

`t`	The reliability option is enabled or disabled.
`nil`	The reliability setup or option with the specified name does not exist.

Examples

Opens a setuplibrary view and sets the status of the reliability options status.

```
slaOpenOrCreateView("Two_Stage_Opamp" "Project45nm" "setupLib")
=> t
slaSetReliabilityOptionsEnabled("ReliabilitySetup1" "Reliability1" nil)
=> t
slaSaveAndCloseView()
=> t
```

Related Topics

slaAddReliabilityOptions

slaSetSimulationSetupRunOptionValue

```
slaSetSimulationSetupRunOptionValue(
     t_setupName
     t_optionName
     t_optionValue
     )
     => t / nil
```

Description

Sets the value of the given run option for the specified simulation setup from the setup library view that is opened using the `slaOpenOrCreateView` function.

Arguments

`t_setupName`	Specifies the name of the simulation setup.
`t_optionName`	Specifies the name of the run option.

The supported run options are as follows:

- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcnumpoints" "11")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "samplingmode" "random")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "samplingmode" "lhs")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "samplingmode" "lds")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcmethod" "all")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcmethod" "process")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcmethod" "mismatch")`
- `slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "montecarloseed" "1111")`

`t_optionValue`	Specifies the new value for the run option.

Value Returned

`t`	The value of the run option is changed to *`t_optionValue`*
`nil`	The simulation setup with the specified name does not exist, or the command failed.

Examples

Opens a setup library view and sets the value of run option for the specified simulation setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaSetSimulationSetupRunOptionValue("SimpleMC_10Points" "mcnumpoints" "10")
=> t
```

Returns an error message if the specified simulation setup does not exist.

```
slaSetSimulationSetupRunOptionValue("XXX" "mcnumpoints" "10")
ERROR (ADE-11414): Cannot run the function because the simulation setup 'XXX' is
invalid. Specify a valid simulation setup name and rerun the function.
=> nil
```

Related Topics

slaAddCornerModelFile

slaAddCornerVariable

slaAddDocument

slaAddSweepVariable

slaImportCorners

slaImportSweeps

Setup Library Assistant Functions

slaUngroupParametricSet

```
slaUngroupParametricSet(
     g_setupName
     g_setId
     )
     => t / nil
```

Description

Ungroups the parametric set from a sweep setup in the setup library view that is opened using the `slaOpenOrCreateView` function in edit mode.

Arguments

g_setupName	Specifies the name of the sweep setup.
g_setId	An integer or string number to denote the parametric set ID. For example, 1 or "1".

Value Returned

`t`	The parametric set is ungrouped.
`nil`	The sweep setup or parametric set with the specified name does not exist, or the command failed.

Examples

Opens a setup library view in edit mode.

```
slaOpenOrCreateView("test" "ProjectSetup" "setupLib")
=> t
```

Returns `nil` if invalid arguments are specified while ungrouping a parametric set.

```
slaUngroupParametricSet("SweepSetup1" "XXX")
ERROR (ADE-11414): Cannot run the function because the parametric set 'XXX' is
invalid. Specify a valid parametric set name and rerun the function.
=> nil
```

Returns `t` if a valid argument is specified while ungrouping variables from a parametric set.

```
slaUngroupParametricSet("SweepSetup1" 0)
=> t
```

Related Topics

slaAddToParametricSet

slaAreParametricSetsEnabled

slaGetParametricSets

slaGroupAsParametricSet

slaRemoveFromParametricSet

slaSetParametricSetsEnabled

Setup Library Assistant Functions

7 Snapshot Functions

Verifier saves snapshots in a zipped format in the `snapshots` directory of a Verifier cellview. These snapshots of your verification progress contain status information on a date and time. You can track the progress over these timestamps and compare results at different stages in the verification cycle.

Snapshots also serve as a time-stamped backups of your setup. Use these backups to restore your active setup to a previous state. By taking a snapshot of your verification project at early stages, you can ensure that all requirements are the same and prevent unintended changes. You can also verify that results, before and after major software, environment or PDK changes, are the same.

The following SKILL functions let you manage snapshots in the ADE Verifier environment.

- verifAreSnapshotsEnabled
- verifCompareImplementationsFromSnapshot
- verifCreateSnapshot
- verifCreateSnapshotConfiguration
- verifDeleteSnapshot
- verifDeleteSnapshotConfiguration
- verifExportSnapshotsToExcel
- verifGetReferenceSnapshot
- verifGetSnapshot
- verifGetSnapshotAbsoluteTolerance
- verifGetSnapshotComment
- verifGetSnapshotRelativeTolerance
- verifGetSnapshots

- verifGetSnapshotsData
- verifIsSnapshotLocked
- verifIsSnapshotVisible
- verifRenameSnapshot
- verifRestoreFromSnapshot
- verifSetReferenceSnapshot
- verifSetSnapshotAbsoluteTolerance
- verifSetSnapshotComment
- verifSetSnapshotConfiguration
- verifSetSnapshotLocked
- verifSetSnapshotRelativeTolerance
- verifSetSnapshotVisible
- verifSetSnapshotsEnabled

Related Topics

Requirement Functions

Implementation Functions

Simulation and Results Extraction Functions

verifAreSnapshotsEnabled

```
verifAreSnapshotsEnabled(
     g_sessionId
     )
     => t / nil
```

Description

Checks if snapshots are enabled in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Snapshots are enabled in the specified Verifier session.
`nil`	Snapshots are disabled in the specified Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a new cellview.

```
winId = deNewCellView("test" "snapshots" "verifier" "verifier" nil)
INFO (VERIFIER-8215): Started Verifier session '0'.
window:2
uid = winId->verifSession
=> 0
```

Checks if snapshots are enabled. By default, the snapshots are disabled.

```
verifAreSnapshotsEnabled(uid)
=> nil
```

Enable the snapshots.

```
verifSetSnapshotsEnabled(uid t)
=> t
verifAreSnapshotsEnabled(uid)
=> t
```

Related Topics

verifSetSnapshotsEnabled

Snapshot Functions

verifCompareImplementationsFromSnapshot

```
verifCompareImplementationsFromSnapshot(
     g_sessionId
     t_srcSnapshotName
     t_dstSnapshotName
     [ g_showDifferences ]
     )
     => t / nil / l_comparisonResults
```

Description

Compares the differences between implementations from snapshots. The implementation comparison targets include the ADE Assembler setup, design (schematic or configurations) and simulation files (model or definition files). You can view the differences in a form or retrieve the list of differences through this SKILL function

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_srcSnapshotName	Name of the source snapshot.
t_dstSnapshotName	Name of the destination snapshot. This snapshot is compared to the reference snapshot.
g_showDifferences	Shows differences in a form. By default, returns a list of differences.

Value Returned

`t`	Shows the implementation differences in a form.
`nil`	Differences cannot be compared because of errors.
l_comparisonResults	List of implementation comparison results between the two snapshots.

Examples

The following example opens a Verifier cellview and compares the implementations from the snapshots.

Start with setting the `includeImplementations` environment variable.

Returns the identical elements.

```
verifCompareImplementationsFromSnapshot(0 "active" "snap_1")
((nil Two_Stage_Opamp\/OpAmp\/maestro_basic\/Active
    ("Identical" "Identical")
    )
)
```

Modifies the maestro cellview, runs the simulation, and then, does the comparison.

```
verifCompareImplementationsFromSnapshot(0 "active" "snap_1")
((nil Two_Stage_Opamp\ / OpAmp\ / maestro_basic\ / Active((nil Common(nil
history("verifier_srsc_20210907_101348.0.RO"
"verifier_srsc_20210907_100655.0.RO") name("Two_Stage_Opamp/OpAmp/maestro_basic/
Active" "Two_Stage_Opamp/OpAmp/maestro_basic/Active")))
    (nil Assembler\ setup(nil setupdb\[historyData\]
            (nil history\[History\]
                    (nil historyentry\[historyData\]
                            (nil checkpoint(nil specs(nil spec\[AC\.Gain\]
                                    (nil min\[47\]
                                            (nil value("47" "44"))
                                        )
                                )) localpsfdir\[\/tmp\/
shbjlnxade2_lifang_101350705\]
                                        (nil value("/tmp/
shbjlnxade2_lifang_101350705"
                                                "/tmp/
shbjlnxade2_lifang_100658031")) timestamp\[Sep\ 7\ 10\: 13\: 50\ 2021\]
                                        (nil value("Sep 7 10:13:50 2021"
                                                "Sep 7 10:06:58 2021"))
                                ))
                        ) statedb\[verifier_srsc_20210907_101348\ .0\.RO\]
                        (nil Test(nil component\[asiEnvSaveTable\]
                                (nil partition(nil field\[\"\"\]
                                                (nil value("\"\""
                                                        "\"10\""))
                                        )) component\[outputs\]
                                        (nil partition(nil field(nil
field\[sevOutputStruct\]
                                                (nil value("-"
                                                                "remove"))
                                            )))
                                    ) modifiedTime("09/07/2021 10:13:51"
                                            "09/07/2021 10:06:59")
value("verifier_srsc_20210907_101348.0.RO"
                                            "verifier_srsc_20210907_100655.0.RO")
                        ))
                )
                (nil Design\\(schematic\ / configurations\)
                        ("Identical"
                                "Identical")
                )
                (nil Simulation\ files\\(model\ / definition\ files\)
                        ("Identical"
                                "Identical")
                )
        ))
)
```

Related Topics

verifAreSnapshotsEnabled

Snapshot Functions

verifCreateSnapshot

```
verifCreateSnapshot(
     g_sessionId
     [ t_snapshotName ]
     [ ?name t_name ]
     [ ?prefix t_prefix ]
     [ ?comment t_comment ]
     [ ?visible g_visible ]
     [ ?locked g_locked ]
     )
     => t / nil
```

Description

Creates a new snapshot '`*.snapz`' that contains the following:
- An archive, stored in `.json` format, of the current requirements, mappings, and results as shown in *Results* tab.
- A copy of the `settings.v3` setup.
- A copy of the results directory.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot. By default, the name is in the `'snap_<date and time>'` format. This argument cannot be used with the `?name` argument.
`?name` *t_name*	Name of the snapshot. By default, the name is in the `'snap_<date and time>'` format. This argument cannot be used with the *tSnapshotName* argument.
`?prefix` *t_prefix*	Prefix of the snapshot. By default, the prefix is `'snap_'`.
`?comment` *t_comment*	Comment added to the snapshot.
`?visible` *g_visible*	Visibility status of the snapshot.
`?locked` *g_locked*	Locked status of the snapshot.

Value Returned

`t`	Snapshot is created successfully in the specified Verifier session.
`nil`	Snapshot does not exist or Verifier does not have the permission to create a new snapshot, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and create a new snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Creates a snapshot with the default naming format.

```
verifCreateSnapshot(uid)
=> t
```

Creates a snapshot with specified name.

```
verifCreateSnapshot(uid "mySnap")
=> t
```

Creates a snapshot automatically.

```
verifRegisterCallback(
    lambda((sess sig args)
            when(sig == 'simulationsAdded
                verifSetSnapshotsEnabled(sess t)
                verifCreateSnapshot(sess ?prefix "sim" ?comment "Pre-simulation 
snapshot")
        )
    )
)
```

Related Topics

verifAreSnapshotsEnabled

verifCreateSnapshotConfiguration

verifDeleteSnapshot

verifSetSnapshotsEnabled

Snapshot Functions

verifCreateSnapshotConfiguration

```
verifCreateSnapshotConfiguration(
     g_sessionId
     t_configName
     l_properties
     [ ?absoluteTolerance x_absoluteTolerance ]
     [ ?relativeTolerance x_relativeTolerance ]
     [ ?differences g_differences ]
     [ ?compareVisible g_compareVisible ]
     [ ?hideEmpty g_hideEmpty ]

     )
     => t / nil
```

Description

Creates snapshot configuration that contains the setup for *Abs Tolerance*, *Rel Tolerance* and *Show* filters on the *Snapshots* tab.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_configName	Name of the configuration.
l_properties	List of requirement properties that are shown on the snapshots tree.

`?absoluteTolerance` *`x_absoluteTolerance`*

Integer or floating-point number specifying absolute tolerance.

`?relativeTolerance` *`x_relativeTolerance`*

Integer or floating-point number specifying relative tolerance.

`?differences` *`g_differences`*

Shows the difference in property values, if enabled. The default is `nil`.

`?compareVisible` *`g_compareVisible`*

Compares the visible property values, if enabled. The default is `nil`.

`?hideEmpty` *`g_hideEmpty`*

Hides the empty property values, if enabled. The default is `nil`.

Value Returned

`t`	Snapshot configuration is created successfully.
`nil`	Command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and creates a new snapshot configuration.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Creates a snapshot configuration and show the properties.

```
verifCreateSnapshotConfiguration(uid "config" list("Title" "Type"
"OverallStatus"))
```

Creates a snapshot configuration and only show different properties.

```
verifCreateSnapshotConfiguration(uid "diff" list("Title" "Type" "OverallStatus")
?differences t)
```

Related Topics

verifDeleteSnapshotConfiguration

verifSetSnapshotConfiguration

Snapshot Functions

verifDeleteSnapshot

```
verifDeleteSnapshot(
     g_sessionId
     t_snapshotName
     )
     => t / nil
```

Description

Deletes the snapshot from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.

Value Returned

`t`	Snapshot is deleted in the specified Verifier session.
`nil`	Snapshot is not deleted because it does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and deletes the snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Deletes the snapshot with specified name.

```
verifGetSnapshots(uid)
("active" "snap_2019_07_08_11_25_26" "mySnap")
verifDeleteSnapshot(uid "snap_2019_07_08_11_25_26")
=> t
verifGetSnapshots(uid)
("active" "mySnap")
```

Returns `nil` if a call is made to delete the `active` snapshot. The `active` snapshot cannot be deleted.

```
verifDeleteSnapshot(uid "active")
=> nil
```

Related Topics

verifCreateSnapshot

verifDeleteSnapshotConfiguration

verifRenameSnapshot

verifRestoreFromSnapshot

Snapshot Functions

verifDeleteSnapshotConfiguration

```
verifDeleteSnapshotConfiguration(
     g_sessionId
     t_configName
     )
     => t / nil
```

Description

Deletes the snapshot configuration with the specified name.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_configName	Name of the configuration.

Value Returned

`t`	Snapshot configuration is deleted successfully in the specified Verifier session.
`nil`	Snapshot configuration does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and deletes the snapshot configuration.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Deletes the snapshot configuration '`diff`'.

```
verifDeleteSnapshotConfiguration(uid "diff")
=> t
```

Deletes a nonexistent snapshot configuration.

```
verifDeleteSnapshotConfiguration(uid "config")
*WARNING* (VERIFIER-5006): verifDeleteSnapshotConfiguration : Cannot delete
snapshots configuration 'config' as it does not exist in the current session.
=> nil
```

Related Topics

verifCreateSnapshotConfiguration

verifDeleteSnapshot

verifSetSnapshotConfiguration

Snapshot Functions

verifExportSnapshotsToExcel

```
verifExportSnapshotsToExcel(
     g_sessionId
     t_fileName
     g_exportAll
     )
     => t / nil
```

Description

Exports the snapshot to the specified MS Excel file.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	Name of the MS Excel file.
g_exportAll	Boolean to specify whether all the snapshot data or only visible properties need to be exported to the MS Excel file.

Value Returned

`t`	Export of snapshots to MS Excel file is successful.
`nil`	The command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and exports snapshots to an MS Excel file.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Export the visible snapshots data with format to Excel file.

```
verifExportSnapshotsToExcel(uid "mySnapshots")
INFO (VERIFIER-10019): Beginning export of snapshots to Excel file
'mySnapshots.xlsx'...
=> t
INFO (VERIFIER-10020): Successfully exported snapshots to Excel file
'mySnapshots.xlsx'.
```

Export all the snapshots data with format to Excel file.

```
verifExportSnapshotsToExcel(uid "mySnapshots_all" t)
INFO (VERIFIER-10019): Beginning export of snapshots to Excel file
'mySnapshots.xlsx'...
=> t
INFO (VERIFIER-10020): Successfully exported snapshots to Excel file
'mySnapshots.xlsx'.
```

Related Topics

verifAreSnapshotsEnabled

verifCompareImplementationsFromSnapshot

verifExportSnapshotsToExcel

verifGetSnapshotsData

verifRestoreFromSnapshot

Snapshot Functions

verifGetReferenceSnapshot

```
verifGetReferenceSnapshot(
     g_sessionId
     )
     => t_refSnapshotName / nil
```

Description

Retrieves the reference snapshot from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

t_refSnapshotName	Name of the reference snapshot.
`nil`	Command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and retrieves the reference snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Retrieves the snapshot with the specified name.

```
verifGetReferenceSnapshot(uid)
"active"
```

Related Topics

verifGetReferenceSnapshot

verifGetSnapshot

verifGetSnapshots

verifGetSnapshotsData

verifSetReferenceSnapshot

Snapshot Functions

verifGetSnapshot

```
verifGetSnapshot(
     g_sessionId
     t_snapshotName
     )
     => l_data / nil
```

Description

Retrieves the disembodied property list for snapshot from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.

Value Returned

l_data	Disembodied property list for snapshot.
`nil`	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and retrieves the disembodied property list for snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifGetSnapshot(uid "snap_2019_07_08_11_25_26")
(nil comment "" customProperties (nil)
date "Jul 8 14:57:32 2019" locked nil name
"snap_2019_07_08_11_25_26" requirements nil user "tom"
visible t
)
```

Related Topics

verifGetSnapshots

verifGetSnapshotsData

Snapshot Functions

verifGetSnapshotAbsoluteTolerance

```
verifGetSnapshotAbsoluteTolerance(
     g_sessionId
     )
     => f_value / nil
```

Description

Returns the absolute tolerance value used when comparing snapshot values.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

f_value	Floating point value of the tolerance.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the tolerance.

```
sess = verifOpenCellView("mylib" "mycell" "verifier")
=> 0
verifGetSnapshotAbsoluteTolerance(sess)
=> 1.34
```

Related Topics

verifGetSnapshotRelativeTolerance

verifSetSnapshotAbsoluteTolerance

verifSetSnapshotRelativeTolerance

Snapshot Functions

verifGetSnapshotComment

```
verifGetSnapshotComment(
     g_sessionId
     t_snapshotName
     )
     => t_comment / nil
```

Description

Returns the comment text for a snapshot.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.

Value Returned

t_comment	Comment text.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the comments for a snapshot.

```
sess = verifOpenCellView("mylib" "mycell" "verifier")
=> 0
verifGetSnapshotComment(sess "snap1")
=> "The first snapshot"
```

Related Topics

verifSetSnapshotComment

Snapshot Functions

verifGetSnapshotRelativeTolerance

```
verifGetSnapshotRelativeTolerance(
     g_sessionId
     )
     => f_value / nil
```

Description

Returns the relative tolerance value used when comparing snapshot values.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

f_value	Floating point value of the tolerance.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the tolerance.

```
sess = verifOpenCellView("mylib" "mycell" "verifier")
=> 0
verifGetSnapshotRelativeTolerance(sess)
=> 2.54
```

Related Topics

verifGetSnapshotAbsoluteTolerance

verifSetSnapshotAbsoluteTolerance

verifSetSnapshotRelativeTolerance

Snapshot Functions

verifGetSnapshots

```
verifGetSnapshots(
     g_sessionId
     )
     => l_snapshots / nil
```

Description

Retrieves the list of snapshots from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_snapshots	List of snapshots.
`nil`	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and retrieves the list of snapshots.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifGetSnapshots(uid)
=> ("active" "snap_2019_07_08_11_25_26" "mySnap")
```

Related Topics

verifGetSnapshot

verifGetSnapshotsData

Snapshot Functions

verifGetSnapshotsData

```
verifGetSnapshotsData(
     g_sessionId
     [ g_showAll ]
     )
     => l_snapshotsData / nil
```

Description

Retrieves the disembodied property list for snapshots from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_showAll	Boolean that controls whether to show all details, that is, all contents of every snapshot.

Value Returned

l_snapshotsData	Disembodied property list for snapshots.
`nil`	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and retrieves a disembodied property list for snapshots.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifGetSnapshotsData(uid)
(nil activeConfig "" configs nil
enabled nil reference "mySnap" snaps
((nil comment "Active requirements snapshot data" date "Jul 8 15:57:46 2019"
locked t name "active" user "tom" visible t
)
(nil comment "" date "Jul 8 13:59:58 2019" location "snapshots/
snap_2019_07_08_11_25_26/snapshot.json" locked nil name "mySnap" user "tom"
visible t)
(nil comment "" date "Jul 8 14:57:32 2019" location "snapshots/
snap_2019_07_08_11_25_26/snapshot.json" locked nil name "snap_2019_07_08_11_25_26"
user "tom" visible t)
)
)
```

Related Topics

verifAreSnapshotsEnabled

verifExportSnapshotsToExcel

verifGetSnapshot

verifGetSnapshots

verifRestoreFromSnapshot

Snapshot Functions

verifIsSnapshotLocked

```
verifIsSnapshotLocked(
     g_sessionId
     t_snapshotName
     )
     => t / nil
```

Description

Returns the locked status of the snapshot.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot to query.

Value Returned

`t`	Snapshot is locked.
`nil`	Snapshot is not locked or the command is not successful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and queries if a snapshot is locked.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifIsSnapshotLocked(sess "snap1")
=> t
verifIsSnapshotLocked(sess "snap2")
=> nil
verifIsSnapshotLocked(sess "invalid-snap")
=> t
*WARNING* (VERIFIER-10003): Cannot find snapshot 'invalid-snap' as there is no
snapshot with this name.
Use a valid name and try again.
=> nil
```

Related Topics

verifSetSnapshotLocked

Snapshot Functions

verifIsSnapshotVisible

```
verifIsSnapshotVisible(
     g_sessionId
     t_snapshotName
     )
     => t / nil
```

Description

Returns the visibility status for the specified snapshot.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.

Value Returned

`t`	Snapshot is visible.
`nil`	Snapshot is not visible or the command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and returns the visibility status of a snapshot.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifIsSnapshotVisible(sess "snap1")
=> t
verifIsSnapshotVisible(sess "snap2")
=> nil
verifIsSnapshotVisible(sess "invalid-snap")
*WARNING* (VERIFIER-10003): Cannot find snapshot 'invalid-snap' as there is no
snapshot with this name.
Use a valid name and try again.
=> nil
```

Related Topics

verifSetSnapshotVisible

Snapshot Functions

verifRenameSnapshot

```
verifRenameSnapshot(
     g_sessionId
     t_snapshotName
     t_newSnapshotName
     )
     => t / nil
```

Description

Renames a snapshot with the specified name.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.
t_newSnapshotName	New name of the snapshot.

Value Returned

`t`	Snapshot is renamed successfully.
`nil`	Snapshot does not exist or the command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and renames the specified snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifRenameSnapshot(uid "snap_2019_07_08_11_25_26" "mySnap_1")
=> t
```

Related Topics

verifCreateSnapshot

verifCreateSnapshotConfiguration

verifDeleteSnapshot

verifRestoreFromSnapshot

Snapshot Functions

verifRestoreFromSnapshot

```
verifRestoreFromSnapshot(
     g_sessionId
     t_snapshotName
     )
     => t / nil
```

Description

Restores a Verifier session from a snapshot along with any stored results.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.

Value Returned

`t`	Verifier session is restored from a snapshot.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and restores a Verifier session from the snapshot.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifRestoreFromSnapshot(sess "snap1")
=> t
```

Related Topics

verifDeleteSnapshot

verifDeleteSnapshotConfiguration

verifExportSnapshotsToExcel

verifGetSnapshotsData

Snapshot Functions

verifSetReferenceSnapshot

```
verifSetReferenceSnapshot(
     g_sessionId
     t_refSnapshotName
     )
     => t / nil
```

Description

Sets the reference snapshot in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_refSnapshotName	Name of the reference snapshot.

Value Returned

`t`	Reference snapshot is set successfully.
`nil`	Command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview `'test/snapshots/verifier'` and sets the reference snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetReferenceSnapshot(uid "mySnap")
=> t
```

Related Topics

verifAreSnapshotsEnabled

verifGetReferenceSnapshot

Snapshot Functions

verifSetSnapshotAbsoluteTolerance

```
verifSetSnapshotAbsoluteTolerance(
     g_sessionId
     f_value
     )
     => t / nil
```

Description

Sets the absolute tolerance for snapshot comparison in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
f_value	Floating point number for the tolerance.

Value Returned

`t`	Absolute tolerance is set successfully.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and sets the absolute tolerance.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetSnapshotAbsoluteTolerance(sess 1.2)
=> t
```

Related Topics

verifGetSnapshotAbsoluteTolerance

verifGetSnapshotRelativeTolerance

verifSetSnapshotRelativeTolerance

Snapshot Functions

verifSetSnapshotComment

```
verifSetSnapshotComment(
     g_sessionId
     t_snapshotName
     t_comment
     )
     => t / nil
```

Description

Sets the comment for a snapshot in a Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.
t_comment	Comment for the snapshot.

Value Returned

`t`	Comment is set successfully.
`nil`	Snapshot does not exist or command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and sets the comment for the specified snapshot.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetSnapshotComment(uid "snap_2019_07_08_11_25_26" "This is a comment")
=> t
```

Related Topics

verifGetSnapshotComment

Snapshot Functions

verifSetSnapshotConfiguration

```
verifSetSnapshotConfiguration(
     g_sessionId
     t_configName
     )
     => t / nil
```

Description

Saves the current *Show* list configuration with the specified name and filters the *Snapshots* tab items with the specified configuration.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_configName	Name of the configuration.

Value Returned

`t`	Current snapshot configuration is set successfully.
`nil`	Snapshot configuration does not exist or command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and sets the current snapshot configuration.

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

Sets the current snapshot configuration to `diff`.

```
verifSetSnapshotConfiguration(uid "diff")
=> t
```

Sets the current snapshot configuration to a non-existent one.

```
verifSetSnapshotConfiguration(uid "config")
*WARNING* (VERIFIER-5006): verifSetSnapshotConfiguration : Cannot set current
snapshot configuration to 'config' as it does not exist in the current session or
it is already the current configuration.
=> nil
```

Related Topics

verifCreateSnapshotConfiguration

verifDeleteSnapshotConfiguration

Snapshot Functions

verifSetSnapshotLocked

```
verifSetSnapshotLocked(
     g_sessionId
     t_snapshotName
     g_isLocked
     )
     => t / nil
```

Description

Locks or unlocks the specified snapshots in the given Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_snapshotName	Name of the snapshot.
g_isLocked	Boolean to specify whether the snapshot should be locked or not.

Value Returned

`t`	Snapshot is locked or unlocked.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and then locks and unlocks a snapshot.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetSnapshotLocked(sess "snap1" t)
=> t
verifSetSnapshotLocked(sess "snap1" nil)
=> t
```

Related Topics

verifIsSnapshotLocked

Snapshot Functions

verifSetSnapshotRelativeTolerance

```
verifSetSnapshotRelativeTolerance(
     g_sessionId
     f_value
     )
     => t / nil
```

Description

Sets the relative tolerance for snapshot comparison in a Verifier session.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`f_value`	Floating point number for the tolerance.

Value Returned

`t`	Relative tolerance is set successfully.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and sets its relative tolerance.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetSnapshotRelativeTolerance(sess 1.2)
=> t
```

Related Topics

verifGetSnapshotAbsoluteTolerance

verifGetSnapshotRelativeTolerance

verifSetSnapshotAbsoluteTolerance

Snapshot Functions

verifSetSnapshotVisible

```
verifSetSnapshotVisible(
     g_sessionId
     t_snapshotName
     g_isVisible
     )
     => t / nil
```

Description

Sets the visibility of a snapshot in a Verifier session. When made visible, it appears as a column in the *Snapshots* tab.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_snapshotName`	Name of the snapshot.
`g_isVisible`	Boolean to specify whether the snapshot should be made visible or not.

Value Returned

`t`	Snapshot is visible or hidden.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview `'test/snapshots/verifier'` and toggles the visibility of a snapshot.

```
sess = verifOpenCellView("test" "snapshots" "verifier")
=> 0
verifSetSnapshotVisible(sess "snap1" t)
=> t
verifSetSnapshotVisible(sess "snap1" nil)
=> t
```

Related Topics

verifIsSnapshotVisible

Snapshot Functions

verifSetSnapshotsEnabled

```
verifSetSnapshotsEnabled(
     g_sessionId
     g_enabled
     )
     => t / nil
```

Description

Enables or disables snapshots in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_enabled	State of 'Enabled'.

Value Returned

`t`	Enables snapshots in the specified Verifier session.
`nil`	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview and changes the 'Enabled' state.

```
winId = deNewCellView("test" "snapshots" "verifier" "verifier" nil)
INFO (VERIFIER-8215): Started Verifier session '0'.
window:2
uid = winId->verifSession
=> 0
```

Returns `nil` if the snapshots are not enabled. By default, snapshots are disabled.

```
verifAreSnapshotsEnabled(uid)
=> nil
```

Enable the snapshots.

```
verifSetSnapshotsEnabled(uid t)
=> t
verifAreSnapshotsEnabled(uid)
=> t
```

Related Topics

verifAreSnapshotsEnabled

Snapshot Functions

8

Verifier-vManager Functions

Verifier lets you set up an analog design verification project using a requirements-driven methodology. From a high level perspective vManager uses the same methodology to let you analyze and plan the verification of your digital requirements. Using Verifier and vManager, you can plan the verification of your project in a mixed-signal environment and analyze all the results together in a single cockpit.

The following SKILL functions let you use the vManager features in the ADE Verifier environment:

- verifCreateVPlan
- verifCreateVsifScript
- verifDownloadFromVManager
- verifGetVManager
- verifGetVManagerProjects
- verifIsVManagerConnected
- verifIsVManagerEnabled
- verifPostResultsToVManager
- verifRemoveVManager
- verifSetVManager
- verifUploadToVManager
- verifVPlanExists

Related Topics

Requirement Functions

verifCreateVPlan

```
verifCreateVPlan(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Creates a vPlan file (`*.vplanx`) for specified Verifier session that is connected to the vManager server.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_fileName`	vPlan file name.

Value Returned

`t`	vPlan file is created successfully.
`nil`	vPlan file exists, or vManager server does not have the permissions to write to that file, or the command is unsuccessful.

Examples

The following example starts a Verifier session with vManager setup and creates a vPlan file.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080
APB_UART_noref.vplanx.
INFO (VERIFIER-8215): Started Verifier session '1'.
=> 0
verifIsVManagerEnabled(uid)
=> t
verifCreateVPlan(uid "/tmp/myPlan.vplanx")
=> t
verifVPlanExists(uid "/tmp/myPlan.vplanx")
=> t
verifCreateVPlan(uid "/home/User/myPlan.vplanx")
*WARNING* (VERIFIER-9001): An error occurred during 'Create Plan':
'Error transferring https://vmgr:letmein@vm-verifier-team:8080/vmgr/vapi/rest/
planning/create - server replied: Bad Request
Failed to open file for writing /home/User/myPlan.vplanx'
```

```
Fix the error and try again.
=> nil
```

Related Topics

verifGetVManagerProjects

verifIsVManagerConnected

verifIsVManagerEnabled

verifVPlanExists

Verifier-vManager Functions

verifCreateVsifScript

```
verifCreateVsifScript(
     g_sessionId
     t_vsifFileName
     [t_batchFileName]
     )
     => t / nil
```

Description

Creates a Verification Simulation Input File (`*.vsif`) and batch script file for the specified Verifier session that is connected to the vManager server. This allows running Verifier from vManager.

Arguments

`g_sessionId`	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
`t_vsifFileName`	VSIF file name.
`t_batchFileName`	Batch script file name that is used to launch Verifier and run simulations. If this value is not defined, the VSIF basename with "_run" appended is used.

Value Returned

`t`	VSIF and batch script files are created successfully.
`nil`	File exists, or Verifier does not have the permissions to write to the file, or the command is unsuccessful.

Examples

The following example starts a Verifier session with vManager setup and creates the required files.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080
APB_UART_noref.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
```

Creates `myRun.vsif` and `myRun_run`.

```
verifCreateVsifScript(uid "myRun.vsif")
=> t
verifCreateVsifScript(uid "scripts/myRun.vsif")
*WARNING* (VERIFIER-5009): Failed to open file 'scripts/myRun_run' for writing.
Check the file permissions and try again.
=> nil
```

Related Topics

verifGetVManagerProjects

verifIsVManagerConnected

verifIsVManagerEnabled

verifUploadToVManager

verifVPlanExists

Verifier-vManager Functions

verifDownloadFromVManager

```
verifDownloadFromVManager(
     g_sessionId
     ?g_waitUntilDone waitUntilDone
     ?t_action action
     )
     => t / nil
```

Description

Downloads the data from specified vPlan file through vManager server.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_waitUntilDone	Ensures that the specified Verifier session waits for the download to complete before proceeding further. This is enabled by default.
t_action	Action for downloading data from vPlan in Verifier. The possible actions are merging the requirements, mapping and implementations from vManager to those in Verifier or replacing them. Specifying 'Ask' displays a dialog to ask what to do. The default value is '`Merge`'. Possible values are '`Ask`', '`Merge`', and '`Replace`'.

Value Returned

`t`	The data is downloaded from the specified vPlan file through vManager server successfully.
`nil`	The command is not successful.

Examples

The following example starts a Verifier session with vManager setup and downloads data.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080
APB_UART_noref.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
```

The vPlan file is modified and the data is downloaded.

```
verifDownloadFromVManager(uid)
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080
APB_UART_noref.vplanx.
=> t
```

Related Topics

verifIsVManagerConnected

verifIsVManagerEnabled

verifPostResultsToVManager

verifUploadToVManager

Verifier-vManager Functions

verifGetVManager

```
verifGetVManager(
     g_sessionId
     [g_addDetails]
     )
     => l_settings / nil
```

Description

Retrieves the vManager settings from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_addDetails	If enabled, the 'rootName' and 'versionSupported' will be shown up in the DPL. By default, only retrieve the basic settings: username, password, project, hostname, port, timeout, vPlan, enabled, version, configured, connected.

Value Returned

l_settings	DPL for the vManager settings in the specified Verifier session.
`nil`	vManager is not configured in the specified Verifier session.

Examples

The following example starts a Verifier session with vManager setup and retrieves the vManager settings.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=>0
verifGetVManager(uid)
(nil username "vmgr" password "***"
project "vmgr" hostname "vm-verifier-team" port
8080 timeout 30 vPlan "/tmp/myPlan.vplanx"
enabled t version "19.01-a001" configured
t connected t
)
verifGetVManager(uid t)
(nil versionSupported t rootName "APB_UART"
username "vmgr" password "***" project
"vmgr" hostname "vm-verifier-team" port 8080
timeout 30 vPlan "/tmp/myPlan.vplanx" enabled
t version "19.01-a001" configured t
connected t
)
```

Related Topics

verifGetVManagerProjects

verifIsVManagerConnected

verifIsVManagerEnabled

verifPostResultsToVManager

verifRemoveVManager

verifSetVManager

Verifier-vManager Functions

verifGetVManagerProjects

```
verifGetVManagerProjects(
     g_sessionId
     )
     => l_projects / nil
```

Description

Retrieves a list of projects from vManager server that are configured in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

l_projects	List of projects from vManager that are configured in the specified Verifier session.
`nil`	vManager is not setup in the specified Verifier session or the command is not successful.

Examples

The following example starts a Verifier session with vManager setup and gets the projects from vManager sever.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=>0
verifGetVManagerProjects(uid)
("vmgr")
```

Related Topics

verifDownloadFromVManager

verifGetVManager

verifIsVManagerConnected

```
verifIsVManagerConnected(
     g_sessionId
     )
     => t / nil
```

Description

Checks if the Verifier session is connected to the vManager server.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	Verifier is connected to the vManager server.
`nil`	Verifier is not connected to the vManager server, or vManager is not fully configured, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and checks if Verifier is connected to the vManager server.

```
uid = verifOpenCellView("test" "example" "verifier")
0
verifIsVManagerConnected(uid)
*WARNING* (VERIFIER-9018): verifIsVManagerConnected - no vManager setup specified
for session 0
Use menu entry Tools->vManager->Setup vManager to configure vManager and try again
nil
```

Related Topics

verifGetVManager

verifIsVManagerEnabled

verifSetVManager

Verifier-vManager Functions

verifIsVManagerEnabled

```
verifIsVManagerEnabled(
     g_sessionId
     )
     => t / nil
```

Description

Checks if vManager is enabled in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	vManager is enabled.
`nil`	vManager is disabled or vManager is not setup in the specified Verifier session.

Examples

The following example starts a Verifier session with vManager setup and checks if vManager is enabled or not.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
```

Returns the check state of 'Enabled' on vManager setup form.

```
verifIsVManagerEnabled(uid)
=> t
```

Related Topics

verifGetVManager

verifIsVManagerConnected

verifSetVManager

Verifier-vManager Functions

verifPostResultsToVManager

```
verifPostResultsToVManager(
     g_sessionId
     [t_vmSessionName]
     )
     => t / nil
```

Description

Posts the verification results of a simulation to the specified vManager session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_vmSessionName	Name of the vManager session that contains all the verification and simulation results for each implementation in Verifier. The default value is `verifier_session_<lib>_<cell>_<view>_<history>_<username>_<time>`.

Value Returned

`t`	Post verification results to the new vManager session successfully.
`nil`	vManager is not setup in the specified Verifier session or the command is not successful.

Examples

The following example starts a Verifier session with vManager setup and posts verification results.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
verifPostResultsToVManager(uid)
*WARNING* (VERIFIER-9014): The implementation 'amsPLL/PLL_VCO_320MHZ_tb/maestro/
Active' has not yet been run by Verifier.
Therefore, no results can be posted to vManager.
```

```
INFO (VERIFIER-9010): Posting results to
verifier_session_amsPLL_TOP_verification_verification_User_2019_03_04_22_15_20_24
10.
INFO (VERIFIER-9011): Successfully post results to
verifier_session_amsPLL_TOP_verification_verification_User_2019_03_04_22_15_20_24
10.
=> t
```

Related Topics

verifDownloadFromVManager

verifGetVManager

verifGetVManagerProjects

verifIsVManagerConnected

verifUploadToVManager

Verifier-vManager Functions

verifRemoveVManager

```
verifRemoveVManager(
     g_sessionId
     )
     => t / nil
```

Description

Remove all the vManager settings from the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.

Value Returned

`t`	All the vManager settings are removed successfully.
`nil`	The command is not successful.

Examples

The following example starts a Verifier session with a cellview and remove the vManager settings.

```
uid = verifOpenCellView("test" "example" "verifier")
=> 0
verifSetVManager(uid ?enabled t ?project "vmgr")
*WARNING* (VERIFIER-9019): vManager setup for session 0 is not fully configured.
Use menu entry Tools->vManager->Setup vManager to fully configure vManager and try
again.
=> t
verifGetVManager(uid)
(nil error "(VERIFIER-9019): vManager setup for session 0 is not fully
configured.\nUse menu entry Tools->vManager->Setup vManager to fully configure
vManager and try again." username ""
password "" project "vmgr" hostname
"" port 0 timeout 30
vPlan "" enabled t version
"" configured nil connected nil
)
verifRemoveVManager(uid)
=> t
verifGetVManager(uid)
*WARNING* (VERIFIER-9018): verifGetVManager - no vManager setup specified for
```

```
session 0
Use menu entry Tools->vManager->Setup vManager to configure vManager and try again
=> nil
```

Related Topics

verifGetVManager

verifIsVManagerConnected

verifIsVManagerEnabled

verifSetVManager

Verifier-vManager Functions

verifSetVManager

```
verifSetVManager(
     g_sessionId
     ?username g_username
     ?password g_password
     ?hostname g_hostname
     ?port g_port
     ?project g_project
     ?timeout g_timeout
     ?vPlan g_vPlan
     ?enabled g_enabled
     ?action t_action
     )
     => t / nil
```

Description

Configures or updates the vManager settings within the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_username	Specifies the user name.
g_password	Specifies the password.
g_hostname	Specifies the host name.
g_port	Specifies the port number.
g_project	Specifies the project name.
g_timeout	Specifies the timeout duration.
g_vPlan	Specifies the vPlan file name.
g_enabled	State of 'Enabled'.
t_action	Action for downloading data from vPlan in Verifier. The possible actions are merging the requirements, mapping and implementations from vManager to those in Verifier or replacing them. Specifying 'Ask' displays a dialog to ask what to do.The default value is '`Merge`'. Possible values are '`Ask`', '`Merge`', and '`Replace`'.

Value Returned

`t`	Configured or modified the vManager settings successfully.
`nil`	Command is unsuccessful.

Examples

The following example starts a Verifier session with vManager setup and modifies the vManager settings.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
verifSetVManager(uid ?enabled nil)
=>t
verifIsVManagerEnabled(uid)
=> nil
```

Creates a new Verifier cellview and configures the vManager settings.

```
win = deNewCellView("amsPLL" "TOP_verification" "verifier" "verifier" nil)
INFO (VERIFIER-8215): Started Verifier session '2'.
window:4
uid = win->verifSession
=> 2
verifSetVManager(uid ?username "vmgr" ?password "***" ?hostname "vm-verifier-team"
?port 8080 ?vPlan "/tmp/myPlan.vplanx" ?enabled t)
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
=> t
```

Updates an existing setup.

```
verifSetVManager(uid ?username "new-user" ?password "newPassword")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
=> t
```

Related Topics

verifGetVManager

verifIsVManagerConnected

verifIsVManagerEnabled

Verifier-vManager Functions

verifUploadToVManager

```
verifUploadToVManager(
     g_sessionId
     [g_waitUntilDone]
     )
     => t / nil
```

Description

Uploads the Verifier data to the specified vPlan file using the vManager server.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
g_waitUntilDone	Ensures that the specified Verifier session must wait for the upload to complete before proceeding further. This is enabled by default.

Value Returned

`t`	Data is uploaded to specified vPlan file through vManager server successfully.
`nil`	Command is not successful.

Examples

The following example starts a Verifier session with vManager setup and uploads data.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=> 0
```

The data is uploaded to vManager from the specified Verifier session.

```
verifUploadToVManager(uid)
INFO (VERIFIER-9005): Uploading session to vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-9006): Successfully uploaded session to vm-verifier-team.8080 /tmp/
myPlan.vplanx.
=> t
```

Related Topics

verifDownloadFromVManager

verifGetVManager

verifGetVManagerProjects

verifPostResultsToVManager

verifSetVManager

Verifier-vManager Functions

verifVPlanExists

```
verifVPlanExists(
     g_sessionId
     t_fileName
     )
     => t / nil
```

Description

Checks if the vPlan file exists and can be accessed by the vManager configured in the specified Verifier session.

Arguments

g_sessionId	Integer, string number, or window specifying the Verifier session ID. For example, `0`, `"0"`, or `window(2)`.
t_fileName	vPlan file name.

Value Returned

`t`	Specified vPlan file exists and is available.
`nil`	Specified Verifier session does not contain vManager setup or the command is not successful.

Examples

The following example starts a Verifier session with vManager setup and checks for the vPlan file.

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=>0
verifVPlanExists(uid "/tmp/myPlan.vplanx")
=>t
verifVPlanExists(uid "myPlan.vplanx")
=>nil
```

Related Topics

verifCreateVPlan

9

Debugging Functions

As a Verifier user, you might experience some problem or unexpected behavior in the tool. Verifier provides a debug mode to let you investigate the root cause of any unexpected behavior or incorrect results.

The issues related to the implementation interface, such as ADE Assembler, might not be visible because they are running in the background when you are working in the ADE Verifier. For debugging such issues, you can run implementations in debug mode, which means that the implementation interface, ADE Assembler, gets displayed for letting you monitor the runs. When you run simulations in the debug mode, the ADE Assembler interface stays open.

The following SKILL functions let you debug issues in the ADE Verifier environment:

- verifEnableDebug
- verifDisableDebug
- verifGetDebug

Related Topics

Requirement Functions

Implementation Functions

Verifier Session and Setup Functions

verifEnableDebug

```
verifEnableDebug(
     [ t_string ]
     )
     => t / nil
```

Description

Enables additional debug logging of various categories.

Arguments

t_string	A string of space-delimited categories that can be enabled. Possible values are: `all`, `db`, `default`, `detailedperformance`, `gui`, `importexport`, `performance`, `simulation`, and `vmanager`. When enabling categories related to performance, the recommendation is to enable the `performance` category instead of `detailedperformance`. Enabling the `detailedperformance` category shows large amount of information, and must be used in rare cases.

Value Returned

`t`	Debugging is successfully enabled.
`nil`	The operation is unsuccessful.

Examples

The following examples describe the possible scenarios.

Enables default debugging category (for example, `db`, `gui`, and `simulation`).

```
verifEnableDebug()
=> t
```

Enables the `db` and `gui` categories.

```
verifEnableDebug("db gui")
=> t
```

Errors out to indicate that the debugging operation has failed because an invalid argument was passed.

```
verifEnableDebug("foo")
*WARNING* (VERIFIER-5022): Cannot enable category 'foo' because it is not a valid
category name. Select a category name from 'all db default detailedperformance gui
importexport performance simulation vmanager' and try again.
=> nil
```

Related Topics

verifDisableDebug

verifGetDebug

Debugging Functions

verifDisableDebug

```
verifDisableDebug(
     [ t_string ]
     )
     => t / nil
```

Description

Disables one or more categories for debug logging.

Arguments

`t_string`	A string of space-delimited categories that can be disabled. Possible values are: `all`, `db`, `default`, `detailedperformance`, `gui`, `importexport`, `performance`, `simulation`, and `vmanager`.

Value Returned

`t`	Debugging is successfully disabled.
`nil`	The operation is unsuccessful.

Examples

The following examples describe the possible scenarios.

Enables default debugging with the `importexport` category (for example, `db`, `gui`, and `simulation`).

```
verifEnableDebug("default importexport")
=> t
```

Disables the `importexport` category.

```
verifDisableDebug("importexport")
=> t
```

Disables the `importexport` category again.

```
verifDisableDebug("importexport")
=> nil
```

Disables debugging by giving no arguments.

```
verifDisableDebug()
verifEnableDebug("foo")
*WARNING* (VERIFIER-5022): Cannot disable category 'foo' because it is not a valid
category name. Select a category name from 'all db default detailedperformance gui
importexport performance simulation vmanager' and try again.
=> nil
```

Related Topics

verifEnableDebug

verifGetDebug

Debugging Functions

verifGetDebug

```
verifGetDebug(
     [ g_all ]
     )
     => t_categories / nil
```

Description

Gets the names of enabled debug categories or all possible categories.

Arguments

g_all	Returns all possible categories if non-zero, else returns only the enabled categories. Possible values are: `all`, `db`, `default`, `detailedperformance`, `gui`, `importexport`, `performance`, `simulation`, and `vmanager`.

Value Returned

t_categories	Returns a space-separated string consisting of all or enabled categories.
`nil`	No categories are enabled.

Examples

The following examples illustrate the possible scenarios.

Gets the currently enabled categories (by default none).

```
verifGetDebug()
=> nil
```

Gets all of the available categories.

```
verifGetDebug(t)
=> "all db default detailedperformance gui importexport performance simulation
vmanager"
```

Enables and then gets those categories.

```
verifEnableDebug("db gui simulation")
=> t
verifGetDebug()
=> "db gui simulation"
```

Disables the `gui` category.

```
verifDisableDebug("gui")
verifGetDebug()
=> "db simulation"
```

Related Topics

verifEnableDebug

verifDisableDebug

Debugging Functions

A

Customization in Functions

You can use Verifier in a top-down, bottom-up, or mixed verification flow for your analog and mixed-signal designs. Verifier lets you set up a verification project with many blocks having multiple verification requirements owned by multiple team members. You can map these requirements to their corresponding simulation setups, which are referred to as implementations. You can simulate the implementations to gather the results. Then, you can review the overall verification status of the entire project, and drill down to the detailed status of each verification requirement in the requirements hierarchy. You can customize Verifier to suit your project. It is possible to reuse your verification setup for other projects to improve the verification quality.

You can use a combination of Verifier SKILL functions to perform complex verification tasks. This topic describes the following examples of SKILL function customization.

- Copy Implementation Units to Requirements
- Callback Function to Save Verifier Cellviews Automatically
- Callback Function to Automatically Create Snapshots for Each Simulation Run
- Callback Function to Automatically Create Snapshots for Backup
- Function to Get Typical Results
- Function to Customize Menu Banners
- Function to Customize Context Menus

Related Topics

Requirement Functions

Implementation Functions

Verifier Session and Setup Functions

Simulation and Results Extraction Functions

Snapshot Functions

Copy Implementation Units to Requirements

The following code illustrates a custom function to get the unit and mapping information from each output of an implementation cellview in a Verifier session. It then sets the unit in the requirements mapped to the output.

This custom function does not copy the specification values set in the implementations to the requirements. To copy the specifications and units of the implementations to the mapped requirements, use the *Override Verifier Specification* feature.

```
Overwrite the unit of requirements that are mapped to the specified implementation
NOTE: This is not support for implementation with 'Run Plan' mode
procedure(copyUnitToReq(sessionId impLib impCell impView impHistory "gt")
    let(((reqIds list()) mappedReqs unit)
        when(verifIsSessionReadOnly(sessionId)
            error("Cannot overwrite the unit of requirement because of read-only
            verifier session")
        )
        Iterate tests in the implementation
        foreach(testName verifGetImpTests(sessionId impLib impCell impView
        impHistory)
            Iterate outputs in the test
            foreach(outputName verifGetImpTestOutputs(sessionId impLib impCell
            impView impHistory testName)
                Get the list of requirements that are mapped to the output
                mappedReqs = verifGetImpMapping(sessionId impLib impCell impView
                impHistory ?testName testName ?outputName outputName)
                when(listp(mappedReqs) && mappedReqs
                    Get the unit of output
                    unit = verifGetImpData(sessionId impLib impCell impView
                    impHistory ?testName testName ?outputName outputName
                    ?dataName "unit")
                    when(stringp(unit) && unit != ""
                        Set the unit for all the mapped requirements
                        foreach(reqId mappedReqs
                            when(verifSetReqProp(sessionId reqId "Unit" unit)
                                reqIds = append1(reqIds reqId)))))))
        reqIds))
```

Copy the above code and paste it in Virtuoso CIW to register the procedure. Then, run the following function in Virtuoso CIW.

copyUnitToReq(*sesssionId impLib impCell impView impHistory*)

For example:

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
copyUnitToReq(sess "")
```

Related Topics

Requirement Functions

Implementation Functions

Simulation and Results Extraction Functions

Callback Function to Save Verifier Cellviews Automatically

The following code illustrates a custom callback function that saves the active Verifier cellviews periodically. You can add such a function in your `.cdsinit` file to save your cellviews automatically.

```
   Callback function to save all the active Verifier cellviews automatically.
   Specify the save interval in seconds as argument, like:
         autoSaveVerifierCellView(300)
   This will save all opened Verifier cellviews every 5 minutes.
   To interrupt the auto saving, set the global variable
         _stopAutoSaveVerifier = t
procedure(autoSaveVerifierCellView(saveInterval "x")
       Get all opened Verifier sessions
    foreach(sessionId verifGetAllSessions()
        ;; Don't save for the read-only session
        unless(verifIsSessionReadOnly(sessionId)
            verifSaveSession(sessionId)))


    if((boundp('_stopAutoSaveVerifier) && _stopAutoSaveVerifier) then
        printf("Auto save interrupted. Set _stopAutoSaveVerifier = nil and start
       autoSaveVerifierCellView %d again.\n" saveInterval)
    else
        hiRegTimer(sprintf(nil "autoSaveVerifierCellView(%d)" saveInterval)
        saveInterval*10)
        printf("...done. Next auto save scheduled in %d seconds.\n"
        saveInterval)))
```

Related Topics

Requirement Functions

Implementation Functions

Verifier Session and Setup Functions

Simulation and Results Extraction Functions

Snapshot Functions

Callback Function to Automatically Create Snapshots for Each Simulation Run

The following code illustrates a custom callback function that automatically creates a snapshot for every simulation. You can add such a function in your `.cdsinit` file to automatically create a snapshot of the session before the simulation starts.

```
Callback function to save create snapshots automatically for each simulation run.
verifRegisterCallback(
    lambda((sess sig args)
        when(sig == 'simulationsAdded
            verifSetSnapshotsEnabled(sess t)
            verifCreateSnapshot(sess ?prefix "sim" ?comment "Pre-simulation 
snapshot")
        )
    )
)
```

Related Topics

Verifier Session and Setup Functions

Simulation and Results Extraction Functions

Snapshot Functions

Callback Function to Automatically Create Snapshots for Backup

The following code illustrates a custom callback function that automatically creates a snapshot for backup purposes. You can add such a function in your `.cdsinit` file to automatically create a snapshot of the session.

```
;; Callback function to save create snapshots automatically for each simulation
run.
;; Callback function to auto save all active Verifier cellviews.
;; Specify the save interval in seconds as argument, like:
;;    AutoCreateSnapshotForCellView("test" "run" "verifier")
;;    AutoCreateSnapshotForCellView("test" "run" "verifier")
;;    AutoCreateSnapshotForSession(sessionId 120)
;;    AutoCreateSnapshotForSession(sessionId)
;;    AutoCreateSnapshot(120 hiGetCurrentWindow())
;;    AutoCreateSnapshot(120)
;;    AutoCreateSnapshot()
;; This will auto create the snapshot for that cellview every 5 minutes.
;; To interrupt the auto creation, set the global variable
;;    StopCreateSnapshot = t
;;
procedure(AutoCreateSnapshotForCellView(lib cell view @optional (interval 300)
"tttx")
    let((sess)
        when(verifIsValidSession(sess = verifGetCellViewSession(lib cell view))
            printf("Starting to create snapshot automatically...\n")
            when(verifCreateSnapshot(sess)
                printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
            )
            if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
            hiRegTimer(sprintf(nil "AutoCreateSnapshotForCellView(\"%s\" \"%s\"
\"%s\" %d)" lib cell view interval) interval*10)
                printf("Next auto creation scheduled in %d seconds.\n" interval)
            else
                printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshotForCellView(<lib>
<cell> <view> 300) again.\n")
            )
        )
    )
)
```

```
procedure(AutoCreateSnapshotForSession(sess @optional (interval 300) "gx")
    when(verifIsValidSession(sess)
        printf("Starting to create snapshot automatically...\n")
        when(verifCreateSnapshot(sess)
            printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
        )
      if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
          hiRegTimer(sprintf(nil "AutoCreateSnapshot(%d)" interval) interval*10)
            printf("Next auto creation scheduled in %d seconds.\n" interval)
        else
            printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshotForSession(<sess>
300) again.\n")
        )
    )
)

procedure(AutoCreateSnapshot(@optional (interval 300) (win hiGetCurrentWindow())
"xg")
    let((sess)
        when(verifIsValidSession(sess = win->verifSession)
            printf("Starting to create snapshot automatically...\n")
            when(verifCreateSnapshot(sess)
                printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
            )
            if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
               hiRegTimer(sprintf(nil "AutoCreateSnapshot(%d)" interval)
interval*10)
                printf("Next auto creation scheduled in %d seconds.\n" interval)
            else
                printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshot(300) again.\n")
            )
        )
    )
)
```

Related Topics

Verifier Session and Setup Functions

Simulation and Results Extraction Functions

Snapshot Functions

Function to Get Typical Results

The following code illustrates a custom function that automatically retrieves the typical results from the specified cellview.

```
procedure(ReadResNum(res map name)
    let((resVal out)
        resVal = ReadRes(res map name)
        when(resVal
            sscanf(resVal "%f" out)
        )
        out
    )
)

procedure(ReadRes(res map name)
    res[car(mapcar(lambda((x) buildString(x "/" )) map))][name]
)

let((reqIDs res map min max typ typParam props sess)
    ;;sess = verifOpenCellView("amsPLL" "TOP_verification" "verification"
?openWindow t)
    sess = 0

    reqIDs = verifGetReqs(sess)
    foreach(reqID reqIDs
        props =  verifGetReqProps(sess reqID)
        printf("\n%s" props["HierId"])
        res = verifGetResultDataForReq(sess reqID)
        map = verifGetReqMapping(sess reqID)
        printf(" -- \t%L  \t"
            reqID
            verifGetReqProp(sess reqID "Title")
            ;;verifGetReqProp(sess reqID "MinSpec")
            ;;verifGetReqProp(sess reqID "MaxSpec")
        )
        when((map && length(map)==1)
            ;; explude mulit mapping
            ;;min = ReadResNum(res map "min")
            ;;max = ReadResNum(res map "max")
            typ = ReadRes(res map "typical")
            if(typ && stringp(typ) && strcmp(typ "various")==0 then
                printf("typical: various ")
              else
                typ = ReadResNum(res map "typical")
                typParam = ReadRes(res map "typicalParams")
                when(typ
                    printf("typical: %L @ %L " typ typParam )
                )
            )
        )
    )
    printf("\n" t)

)
```

Related Topics

Requirement Functions

Verifier Session and Setup Functions

Simulation and Results Extraction Functions

Snapshot Functions

Function to Customize Menu Banners

The following code illustrates menu customization in Verifier. The customization can be placed in different locations. See Customizing the Menu Banner.

Menu files are read in the following order:

```
your_install_dir/etc/tools/menus/appName.menus
your_install_dir/local/menus/appName.menus
workOrProjectArea/menus/appName.menus
~/menus/appName.menus
```

For example, you can place the attached file in `workOrProjectArea/menus/verifier.menus` and start a new Virtuoso session and launch Verifier. You find a new custom menu entry with two actions.

```
;; Verifier menu customization example
verifier.menus:

;; Load the existing Verifier menus in order to add to it instead of replacing it
loadi(prependInstallPath("etc/tools/menus/verifier.menus"))
;; Define functions to be called by new menu entries
procedure(VerifPrintSess(sess)
    printf("%L\n" sess)
)
procedure(VerifPrintReq(sess)
    let((reqIDs)
        reqIDs = verifGetReqs(sess)
        foreach(reqID reqIDs
            printf("%L -- \t%L  \t%L--%L\n"
                reqID
                verifGetReqProp(sess reqID "Title")
                verifGetReqProp(sess reqID "MinSpec")
                verifGetReqProp(sess reqID "MaxSpec")
            )
        )
    )
)
;; Define the new menu entries
myMenu = '(MyMenu "CustomMenu"
    (
        (PrintSess "Print Session Number" "VerifPrintSess(hiGetCurrentWindow()-
>verifSession)")
        (PrintReq "Print Requirements" "VerifPrintReq(hiGetCurrentWindow()-
>verifSession)")
    )
)
;; Add new menus to the banner menus
verifMenus = append1(verifMenus 'myMenu)
```

Related Topics

Verifier Session and Setup Functions

Function to Customize Context Menus

You can define customized context menu entries in ADE Verifier by including any of the following variables in the `verifier.menus` file.

- `verifImpContextMenu`
- `verifReqContextMenu`
- `verifRunContextMenu`
- `verifResultsMenu`

By using any of these environment variables, you can create custom context menus in the *Implementations* or *Requirements* panes of the *Setup* tab or in the *Run* and *Results* tabs. The definitions are added at the end of the respective context menu. For more information, see Function to Customize Menu Banners. You define these variables as a list of items that can be another list, a symbol, or a string in the following format:

(*id text tooltip entry* [*enabled-fn*] [*show-fn*])

Here,

- *id* is the symbol identifier,
- *text* denotes <*menu text*> with its corresponding <*tooltip*>,
- *entry* is either a string with a SKILL callback, a symbol with the name of a function, or a list defining a sub-menu.
- *enabled-fn* and *shown-fn* are optional callbacks that specify if the items should be enabled and shown.

For example, you can place the following code in `work` Or `ProjectArea/menus/verifier.menus` and launch Verifier in a new Virtuoso session. You then find the new menu actions added to the context menus.

```
; The menu entry is a list of items that is a list, a string, or a symbol.
;
; If it is a string, a separator is placed at the menu entry location, and the
; contents of the string are ignored.
;
; If it is a symbol, then the menu entry is a reference to an existing menu entry/
; definition that can be defined elsewhere(for example, as myAbc and mySlider).
;
; If it is a list, then a menu action is created using the contained definition,
; which needs to be in the following format:
;
; (id menu-name menu-tooltip menu-entry menu-tooltip [menu-enabled-fn]
; [menu-item-shown-fn]).
;
; menu-name is the text that is displayed in the menu, and menu-tooltip is its
; corresponding tooltip.
;
; The menu-entry can be one of the following:
;   a) a string: It is evaluated as a callback and no arguments are passed on
;      to the callback.
;   b) a symbol: It is the name of a callable object (i.e. function-name or
;      lambda function) which takes two arguments - session, and selected.
;   c) a list: It defines the contents of a pull-right menu.
;
; The menu-enabled-fn is either nil, a string or a callable function which if
; non-nil, is evaluated to determine if the menu is enabled or not.
;
; The menu-item-shown-fn is either nil, a string or a callable function. If it is
; non-nil, it is evaluated to determine if the menu items are shown in the context
; menu or not. For example, you can use it to only show an item if there is at
; least one item selected, or only show an entry if a single implementation is
; selected .
;
; If the menu or the enabled callback function are not defined, or if the enabled
; function returns an error, then the menu is disabled.
;
; The menu-callback, enable-callback, and show-callback functions should take the
; following two arguments:
           a session-number + list-of-selected-items.
;

info("Loading custom menus...\n")

custom1 = '(custom1 "My Impl 1" "Custom 1 Tooltip" doImpl1CB1)

;; Defines the custom context menu for the Implementations pane in the Setup tab.
verifImpContextMenu =
'(
    custom1
    (custom2 "My Impl 2" "Do more custom stuff" "doImplCB2()" doImplCB2Enabled
isCustom2Shown)
    "----" ; separator
    (slider1 "Slider" "Slide right for extra stuff"
        (
            (item1 "Item1" "Item1 Tooltip" doImplCB1 item1Enabled)
         )
         isSlider1Enabled
    )
    slider2
)
```

```
slider2 = '(slider2 "Slider 2" "Yet another Slider"
               ((item3 "Item3" "Item3 Tooltip" doImplCB1))
               nil showSlider2)

;; Defines the custom context menu for the Requirements pane in the Setup tab.
verifReqContextMenu =
'(
    (req1 "My Req Item" "Do req stuff" doReqCB doReqEnabled isReqShown)
)

;; Defines the custom context menu for the run table in the Run tab.
verifRunContextMenu =
'(
    (run1 "My Run Item" "Do run stuff" doRunCB doRunEnabled isRunShown)
)

;; Defines the custom context menu for the results table in the Results tab.

verifResultsContextMenu =
'(
  (res1 "My Res Item 1" "Do res stuff" doResCB doResEnabled isResShown)
)

info("...finished loading custom menus.\n")
```

Any callback takes two arguments: `(session-number list-of-selected-names).`

The following section describes the various callback functions:

```
; Defines all the various callbacks for the custom menus in the Requirement, Run
; and Results hierarchies
defun( doImplCB1 (sess sel)
    info("doImplCB1 CB session: %L selected set: %L\n" sess sel)
)
defun( doImplCB2 ()
    info("doImplCB2 CB called with no arguments")
)
defun( doImplCB2Enabled (sess sel)
    ;; custom2 is enabled only if more than 2 items are selected.
    length(sel)>2
)
defun( isImpl2Shown (sess sel)
    ;; customer2 is shown only if more than 1 item is selected.
    length(sel)>1
)
defun( item1Enabled (sess sel)
    length(sel)>2
)
defun( isSlider1Enabled (sess sel)
    ;; slider1 is always enabled.
    t
)
defun( showSlider2 (sess sel)
    ;; slider2 is visible only if more than 3 items are selected.
    length(sel)>3
)

defun( doReqCB (sess sel)
    info("Req CB session: %L selected set: %L\n" sess sel)
)
defun( doRunCB (sess sel)
    info("Run CB session: %L selected set: %L\n" sess sel)
)
defun( doResCB (sess sel)
    info("Res CB session: %L selected set: %L\n" sess sel)
)
defun(doReqEnabled (sess sel)
    info("Req Enabled session: %L selected set: %L\n" sess sel)
    t
)
defun(doRunEnabled (sess sel)
    info("Run Enabled session: %L selected set: %L\n" sess sel)
    t
)
defun(doResEnabled (sess sel)
    info("Res Enabled session: %L selected set: %L\n" sess sel)
    t
)
defun(isReqShown (sess sel)
    info("Is Req shown session: %L selected set: %L\n" sess sel)
    t
)
defun(isRunShown (sess sel)
     info("Is Run shown session: %L selected set: %L\n" sess sel)
    t
)
defun(isResShown (sess sel)
    info("Is Res shown session: %L selected set: %L\n" sess sel)
    t
)
```