

Automotive Data Dictionary

ADD MATLAB – Services v21.1.R1

May 07th 2021

ADD Contact

email: hotline@visu-it.com

Internet: visu-it.de/products/add/



© Copyright 2021
Visual Information Technologies GmbH
An der Schergenbreite 1
93059 Regensburg

Contents

1	MATLAB Services	3
1.1	Service ① ADD Export	5
1.1.1	How to start import mask	5
1.1.2	How to use import mask	7
1.1.3	Import ADD	7
1.1.4	Legacy Import	10
1.2	Service ② “vitDDX2WS”	11
1.3	Service ⑤ “vitDDX2MS”	11
1.4	Service ③ “vitWS2DDX”	12
1.5	Service B “vitWS2MS”	12
1.6	Service A “vitADDOffline”	13
1.7	Service “vitCheckMultipleNames”	14
1.8	Service “vitADDCreateNewContainer”	14
1.9	Service “vitDDXvsWS”	15
1.10	Service “vitAccessMetalInfos”	15
1.11	Service “vitDAT2WS” (Trial Version – DCM Only!)	16
1.12	Service “vitWS2DAT” (Trial Version – DCM Only!)	16
2	Questions?	16
3	Appendix	17
3.1	Integrating ADD into MATLAB Simulink	17
3.1.1	Building History	19
3.2	Legacy Import	21
3.3	Structures used by MATLAB Services (Overview)	25

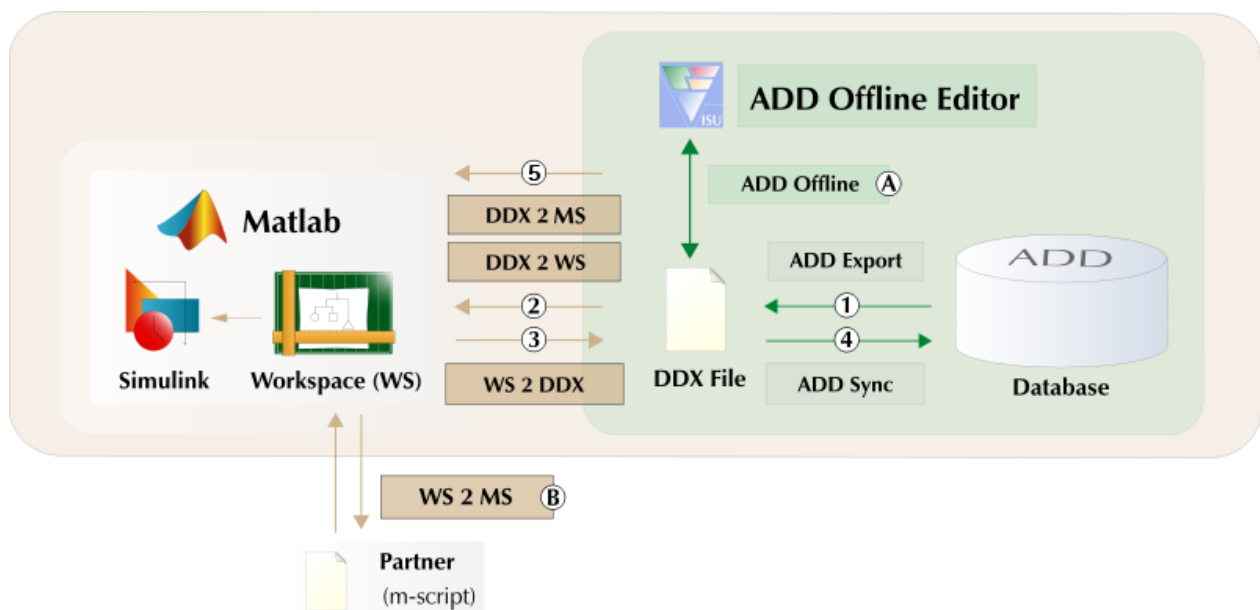
1 MATLAB Services

The task of “MATLAB Services” is to interlace “ADD Offline Editor” with MATLAB Workspace (WS). The lynchpin between the two systems is represented by a ddx – file. This file could be transfused to ADD Database via “ADD Sync” (cf. ④). More information’s about the synchronization tool are available on the “ADD Tool” Help section.

To create a ddx – file out of MATLAB the “ADD Export” must be used. This export is represented in MATLAB by the function “vitADD2MATLAB ()” (cf. ①).

The services “vitDDX2WS” (cf. ②), “vitDDX2MS” (cf. ⑤) and “vitWS2DDX” (cf. ③) represent the gateway between ADD and MATLAB. Furthermore, there is a service “vitWS2MS” (cf. ⑧) implemented which creates an executable MATLAB m – script according to the MATLAB WS and a service “vitADDOffline” (cf. ⑨) that grand access to the ADD Offline Editor.

Hint: Services “vitDDX2WS”, “vitDDX2MS” & “vitWS2DDX” are able to read/write special container files (‘VIT_Ignore_for_ADDS.ddx’) automatically. In order to write WS variables (“vitWS2DDX”) into a special container file, the ‘ignore_for_adds’-attribute of each variable has to be set to value ‘1’. Service “vitDDX2WS” will read the spec. container file ‘VIT_Ignore_for_ADDS.ddx’ automatically, if it’s located in the same directory as your selected ddx – file.



① [ADD Export: vitADD2MATLAB\(\);](#)

③ [WS 2 DDX: vitWS2DDX\(\);](#)

⑧ [WS 2 MS: vitWS2MS\(\);](#)

② [DDX 2 WS: vitDDX2WS\(\);](#)

⑤ [DDX 2 MS: vitDDX2MS\(\);](#)

⑨ [ADD Offline: vitADDOffline\(\);](#)

There is also the possibility to create an empty ddx-file using service “vitADDCreateNewContainer”.

Last but not least there are two more services (“vitDDXvsWS” & “vitCheckMultipleNames”) implemented, designed as check-up functions.

Service “vitCheckMultipleNames” verify the content of given MATLAB WS to make sure that no multiple variable names of struct member could be found. Whereas service “vitDDXvsWS” compares a given MATLAB WS to a ddx-file specified.

In order to use the services, it is necessary to create a WS structure array called “adds”. This “struct” has to include the fields “ddxfile”, “mScript”, “prefix” and “verbose”.

At this junction the structure field

- “**ddxfile**” represents the absolute address of the ddx – file.
MATLAB data type: char array
- “**mScript**” represents the absolute address of the MATLAB m – script (only used by service “vitWS2MS”) .
MATLAB data type: char array
- “**prefix**” represents the prefix of the MATLAB WS variables which will be created or transferred.
MATLAB data type: char array
- “**verbose**” represents a flag to omit or allow progress messages.
MATLAB data type: double array; Values: 0/1

Additional structure field:

- “**add_environment**” represents the name of the actual edited container or project.
MATLAB data type: char array
- “**pinfo**” represents a flag to generate either plane datatype information (=1) or detailed datatype information (=0; create “datatype” and “conversion” structures!).
MATLAB data type: double array; Values: 0/1
- “**ign_param_arrsize_check**” represents a flag to skip array size check for “value” (“Parameter”) objects (=1) or to check the array size definition against the initial dataset (object attribute “values”).
MATLAB data type: double array; Values: 0/1

Hint: Service “vitWS2DDX/MS” will detect the datatype-information-level automatically!

Each service checks first of all the coherence of the “**adds**” – structure.

In case of a non existing structure field

- “**adds.verbose**” : Default will be generated and set to ‘0’.
- “**adds.prefix**” : Default will be generated and set to ‘init_’.
- “**adds.mScript**” : Save dialog appears.
- “**adds.ddxfile**” : Open/Save dialog appears.

Hint: Service “vitADDCreateNewContainer” show up an “Overwrite File?” dialog in case of an already existing ddx – file.

“**adds**” example:

```
adds.ddxfile           = 'C:\Visu-IT!\ADD_Container.ddx'
adds.mScript           = 'C:\Visu-IT!\Interchange.m'
adds.prefix            = 'vit_'
adds.verbose           = 1
adds.add_environment   = 'TEST 1.1.0'
```

All services got two return values which allow an interpretation of the service run. The result of the service run will be stored in the first return value as a logical value ‘1’ or ‘0’.

At this ‘1’ denotes a successful service run and ‘0’ an aborted running. In case of an aborted running the second return value describes the occurred error.

Hint: A warning occurred during service run will return ‘1’ and a not empty error string!

```
e.g.: [result error_string] = vitDDX2WS();
      result                = vitDDX2WS();
```

Hint: Most of the parameter values of the services are optional! If there is no parameter value declaration the respective default value will be used instead.

1.1 Service ① ADD Export

Hint:

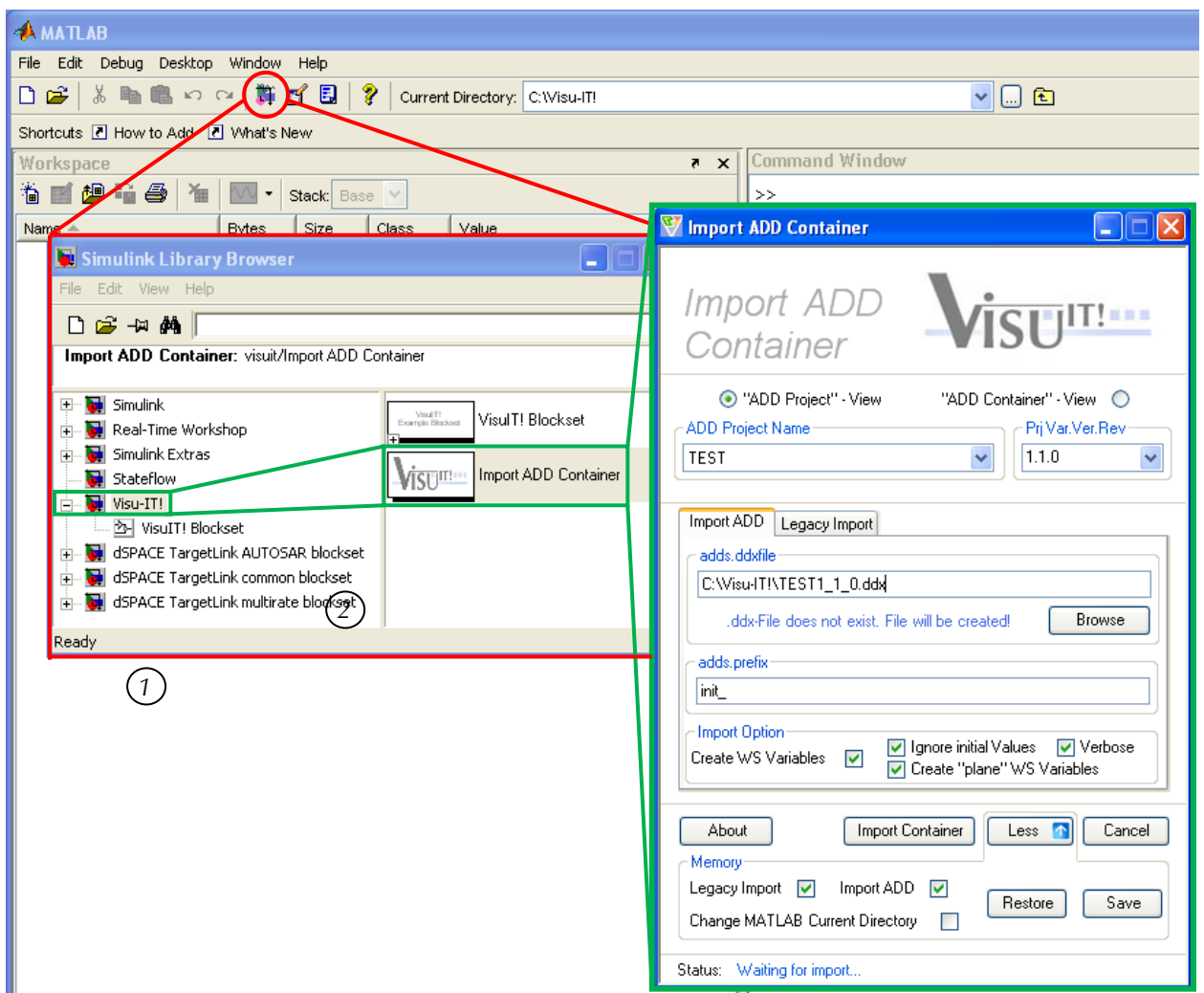
This service is a component of ADD “Online”! In order to use this service, it is necessary to install ADD Database.

The ADD to MATLAB – Interface is used to directly import an ADD Container into MATLAB Workspace. There are two different ways to open the interface.

On one hand the import mask could be started easily from MATLAB Command window on the other hand the interface could be opened by selecting the “Visu-IT!” - block in the MATLAB Simulink Library Browser.

1.1.1 How to start import mask

1. Open “Import ADD Container” using MATLAB Simulink Library Browser:

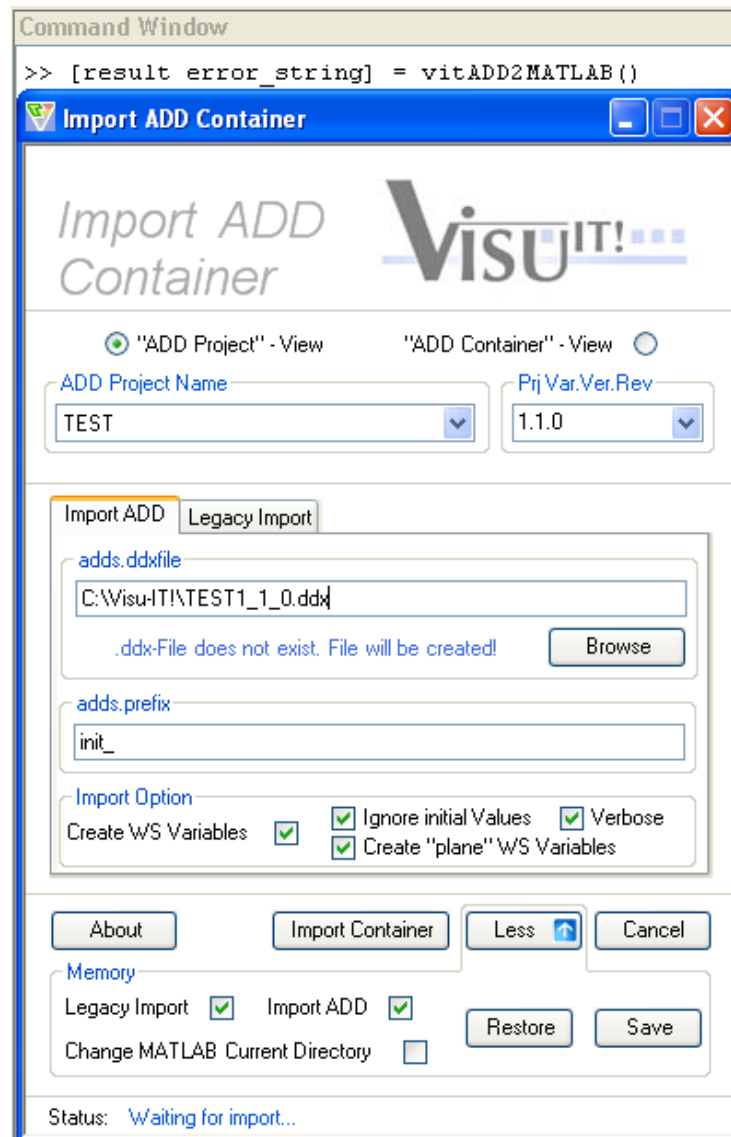


After opening MATLAB Simulink Library Browser the “Visu-IT!” - block (cf. ①) must be selected. By double clicking on the appearing “Import ADD Container” – block (cf. ②) the import mask will be automatically opened.

As a great benefit of this loading style the “Import ADD Container” – block can be handled as any other Simulink block. Hence “Drag & Drop” is possible.

2. Open “Import ADD Container” using MATLAB Command Window:

Service call: `vitADD2MATLAB();`



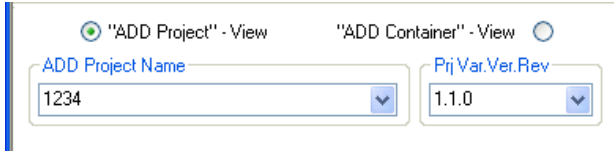
In MATLAB Command Window type “`vitADD2MATLAB()`” to open import interface. To check the quality of the started import it is necessary to define return values.

e.g.: `[result error_string] = vitADD2MATLAB();`
`result = vitADD2MATLAB();`

At this junction `result` denotes a successful (‘1’) or aborted (‘0’) import as a logical value. In case of an aborted run the char array `error_string` describes the error.

1.1.2 How to use import mask

Container/Project section:



The Container/Project – area provides an ADD container –respectively project list based on the ADD Database. By selecting the “ADD Project” – View all available ADD projects will be listed in the “ADD Project Name”

section. By switching to the “ADD Container”-View a list of those container who are linked to the selected project will be generated. By selecting the project “AllCnt”, “ADD Container”-View will display all available containers.

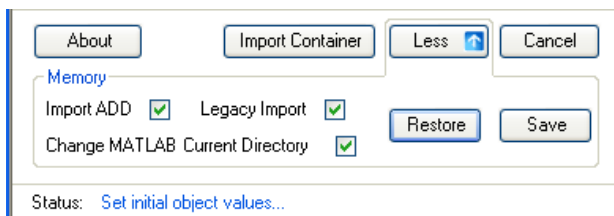
Import style section:

“Import ADD” creates an ADD conform ddx – file. It is also possible to create MATLAB Workspace variables which can be used for “MATLAB Services” (see [Import ADD](#)).

“Legacy Import” creates a MATLAB m – script with an alternative variable structure (see [Legacy Import](#)).

Import and Memory section:

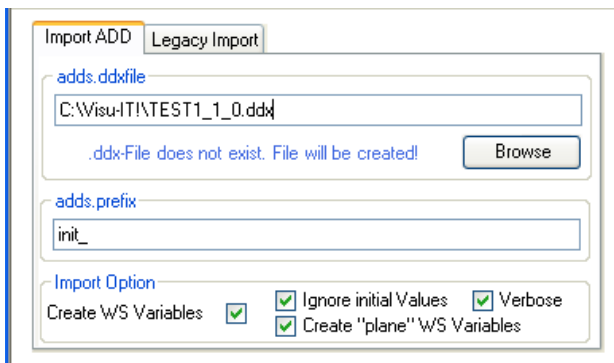
Button “Import Container” starts the ADD container import.



Button “More” expands the “Memory” – area. Choosing the import style and apply with “Save” settings will be stored. To restore the saved settings press button “Restore”.

Setting “Change MATLAB Current Directory” will change the MATLAB Current Directory to the selected file address assigned via “File Path” or “adds.ddxfile”.

1.1.3 Import ADD



DDX - File path section

MATALB variable prefix

Import mode

Import-Mode:

“Create WS Variables”: Generates MATLAB Workspace variables based on the generated ddx – file.

“Ignore Init Values”: Omit the allocation of variable values.

“Create ‘plane’ WS Variables”: Create additional MATLAB Workspace Variables which are containing the ‘value’ content of the appropriate Objects.

“Verbose”: Generate an appropriate MATLAB m – script.

DDX - File path section:

Path and name of the ddx - file which has to be created.

Default path is MATLAB “Current Directory” or the value of the existing MATLAB Workspace variable “adds.ddxfile”.

Default name is the “SRS-Filename” of the selected ADD Container.

MATLAB variable prefix:

Prefix of the MATLAB WS variables which will be created. If there is no MATLAB Workspace variable “adds.prefix” assigned “init_” will be set as default.

“Create WS Variables” amendment:

Name	Value	Size	Bytes	Class
adds	<1x1 struct>	1x1	596	struct
init_axis	<1x1 struct>	1x1	4000	struct
init_map	<1x1 struct>	1x1	2316	struct
vit_ADDaxis_X	[1 2 3 4]	1x4	32	double
vit_ADDaxis_Y	[1 2 3 4 5]	1x5	40	double
vit_ADDmap	<5x4 double>	5x4	160	double

Field	Value
map_name	'vit_ADDmap'
phys_min	-1.7977e+308
phys_max	1.7977e+308
description	'ADD counterpart of element Lookup2D'
unit	''
simdatatype	'double'
conversion	'RATIONALFUNCTION [0;1;0;0;1]'
x_axis_name	'vit_ADDaxis_X'
y_axis_name	'vit_ADDaxis_Y'
category	'grouped'
value	<5x4 double>
ignore_for_adds	'false'
adds_name	'vit_ADDmap'
defined_in_cnt	''
referenced_in_cnt	<0x0 cell>
local_in_cnt	'EMPTY 0.1.0'

Variable structure:

<Prefix><Objekttyp>.<Objektattribute>

- **Prefix:** Prefix of the MATLAB WS variables.
- **Objekttyp:**

MABLAB Denotation	<->	ADD Denotation
value	<->	Parameter
const	<->	cDefine
axis	<->	Axis
curve	<->	Map (1d, Kennlinie)
map	<->	Map (2d, Kennfeld)
measure	<->	Online
diag_idx	<->	Diag_Idx
datatype	<->	Simulation/Software Datatype
conversion	<->	Conversion
codevariant	<->	CodeVariant

• **Objektattribute:**

Structure fields defined for all objects:

< Objekttyp >_name:	Object name
phys_min, phys_max:	Minimum and maximum of the physical values
longName:	User-specific long name of the object
status:	Object status
customtags:	Object custom tag list
unit:	Physical unit
formatstring:	Object formatstring
displayname (*):	Object conversion name
basetype (*):	Object basetype
simdatatype:	Object simulation datatype if (*) else Object basetype
swdatatype (*):	Object software datatype
conversion:	Object conversion
type (*):	Object conversion type
value:	Object values
values (*):	Object conversion values
reference_values:	Object values; Field won't be influenced by argument 'IgnoreInits'!

ignore_for_addds:	Specifies if object should be exported to a special container.
adds_name:	User-specific object name
defined_in_cnt:	List of Container names within the object has been defined.
referenced_in_cnt:	List of Container names within the object has been referenced.
local_in_cnt:	List of Container names within the object is used as a local variable
dcm_function:	Free assignable field
dcm_displayname:	Free assignable field

Additional structure field for object “axis”:

online_reference:	Object’s online reference
maxaxispoints:	Object value count
category:	Axis category
axis_type:	Axis dimension type. Multiple dim. references possible (e.g. x_y_axis)

Additional structure field for object “curve”:

x_axis_name:	Name of the referenced axis
category:	Curve Category

Additional structure field for object “map”:

x_axis_name:	Name of the referenced x - axis
y_axis_name:	Name of the referenced y - axis
category:	Map Category

Additional structure field for object “value”:

is_array_x	x-dimension-specified flag
array_size_x	Object’s x-dimension array size value
size_x_from_system_constant	Object’s x-dimension array size constant name
is_array_y	y-dimension-specified flag
array_size_y	Object’s y-dimension array size value
size_y_from_system_constant	Object’s y-dimension array size constant name

Additional structure fields for object “online”:

bit_carrier	Name of the online carrier object
bit_position	Position definition of the bit object within the carrier
write_only	OnlineWriteOnly flag

Additional structure fields for object “codevariant”:

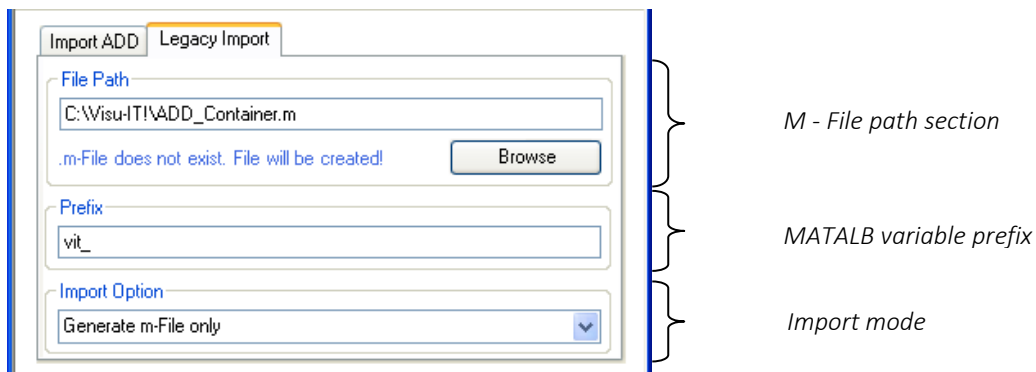
condition	Codevariant condition definition
-----------	----------------------------------

Additional structure fields where object type is **not** “constant”:

visible:	Object’s visible attribute
mem_sec:	Object’s memory section definition
task:	Object’s task definition

(*) The display of this field depends on the value of adds-struct “pinfo”!
pinfo = 0: “basetype”, “displayname”, “swdatatype”, “type” “values” not available!

1.1.4 Legacy Import



M - File path section:

Path and name (without file extension) of the MATLAB m – script which has to be created.
 Default path is MATLAB “Current Directory”. Default name is the “SRS-Filename” of the selected ADD Container.

MATLAB variable prefix:

Prefix of the MATLAB WS variables which will be created.

Import-Mode:

„Generate m-File only“:

Only create an executable MATLAB m - script. (For details see Appendix [[Generate m-File only](#)])

„Import to MATLAB Workspace (incl. m-File!)“:

Create an executable MATLAB m – script and appropriated MATLAB Workspace variables.
 (For details see Appendix [[Import to MATLAB Workspace \(incl. m-File\)](#)])

„Create Container Library (incl. Container Model!)“:

Create a Simulink Library and Model of the selected ADD Container.
 (For details see Appendix [[Create Container Library \(incl. Container Model\)](#)])

1.2 Service ② “vitDDX2WS”

Service call: vitDDX2WS(Verbose, IgnoreInits, pValue);

Alternative: vitDDX2WS(Verbose, IgnoreInits); vitDDX2WS (Verbose); vitDDX2WS();

Based on the existing ddx – file (cf. [adds.ddxfile](#)) the service “vitDDX2WS” (cf. ②) transfuses the ddx – file elements to MATLAB WS. The characteristic of the appropriate variables is the prefix assigned by [adds.prefix](#).

Parameter value	Value	Description
Verbose [optional]	0	Do not create an executable MATLAB m – file. (default)
	1	Create an executable MATLAB m – file representing WS content.
IgnoreInits [optional]	0	Structure field “value” of the WS variables will be set to the appropriate “Phys_Values” of the ddx – file element. If there is no “Phys_Values” available the structure field “value” will be set to appropriate “Phys_min” – value.
	1	Structure field “value” will not be set. (default)
pValue [optional]	0	Do not create additional WS Variables. (default)
	1	Create additional WS Variables which are containing the value content only (plane Variables).

1.3 Service ⑤ “vitDDX2MS”

Service call: vitDDX2MS(IgnoreInits, pValue);

Alternative: vitDDX2MS(IgnoreInits); vitDDX2MS();

Based on the existing ddx – file (cf. [adds.ddxfile](#)) the service “vitDDX2MS” (cf. ⑤) transfuses the ddx – file elements into a MATLAB m - file. The characteristic of the appropriate variables is the prefix assigned by [adds.prefix](#).

Parameter value	Value	Description
IgnoreInits [optional]	0	Structure field “value” of the WS variables will be set to the appropriate “Phys_Values” of the ddx – file element. If there is no “Phys_Values” available the structure field “value” will be set to appropriate “Phys_min” – value.
	1	Structure field “value” will not be set. (default)
pValue [optional]	0	Do not create additional WS Variables. (default)
	1	Create additional WS Variables which are containing the value content only (plane Variables).

1.4 Service ③ “vitWS2DDX”

Service call: vitWS2DDX(Verbose, AutoStartADDOff, IgnoreInits, pValue);
 Alternative: vitWS2DDX(Verbose, AutoStartADDOff, IgnoreInits);
 vitWS2DDX(Verbose, AutoStartADDOff); vitWS2DDX(Verbose); vitWS2DDX();

Based on existing MATLAB WS variables, which containing the surname prefix [adds.prefix](#), the service “vitWS2DDX” (cf. ③) creates an ADD capable ddx – file saved at location [adds.ddxfile](#).

Parameter value	Value	Description
Verbose [optional]	0	Do not create an executable MATLAB m – file. (default)
	1	Create an executable MATLAB m – file representing WS content.
AutoStartADDOff [optional]	0	Omit automatic opening of ADD Offline Editor. (default)
	1	Automatically open the ADD Offline Editor.
IgnoreInits [optional]	0	“Phys_Values” – object of the ddx – file elements will be set to the appropriate WS variable field “value”.
	1	“Phys_Values” – object of the ddx – file elements will not be set. (default)
pValue [optional]	0	Do not create additional WS Variables. (default)
	1	Create additional WS Variables which are containing the value content only (plane Variables). ONLY IF AutoStartADDOff = 1

Hint:

In Case of using the “AutoStartADDOff” – mode the dedicated WS will be updated after closing the ADD Offline editor. If “pValue” set to 1, service also writes additional variables into MATLAB Workspace.

Please notice:

In case of object arrays of type “value” (“Parameter”; “init_values” -> “is_array_x/y/z” = 1) “Phys_Values” must be set; “array_size_x/y/z” or “size_x/y/z_from_system_constant” must match the corresponding dimension of the “Phys_Values”. Otherwise, objects will be skipped.

Check can be deactivated via MATLAB WS variable [adds.ign_param_arrsize_check](#).

1.5 Service ⑥ “vitWS2MS”

Service call: vitWS2MS(limited);
 Alternative: vitWS2MS();


Based on existing MATLAB WS variables, which containing the surname prefix [adds.prefix](#), the service “vitWS2MS” (cf. ⑥) creates an executable MATLAB m – script saved at location [adds.mScript](#).

Parameter value	Value	Description
limited [optional]	‘ignADDS’	All WS variables with struct field value ‘ignore_for_adds’ is set to ‘1’ won’t be written into the new m – script.

1.6 Service “vitADDOffline”

Service call: vitADDOffline(objName, objType, loadMode, adaptDDXFileName);

Alternative: vitADDOffline(objName, objType, loadMode); vitADDOffline(objName, objType);
vitADDOffline(objName); vitADDOffline();

Service “vitADDOffline” (cf ) starts the ADD Offline Editor by loading the ddx – file (adds.ddxfile) and showing its content. If there is no ddx - file assigned in adds.ddxfile a template ddx – file will be generated and opened instead.

Parameter value	Value	Description
objName [optional]	Empty char array	Full access within the ADD Offline Editor. (default)
	char array	Restricted access within the ADD Offline Editor. Only the object assigned in “objName” is allowed to be edited.
objType [optional]	char array	Specifies the type of a newly created ADD object. Hint: “objType” effects only if object with name “objName” does not exist!
loadMode [optional]	0	Full access within the ADD Offline Editor. (default)
	1	MATLAB Simulink Option: For propose of calling the service within MATLAB Simulink Block. Restricted access within the ADD Offline Editor. Only the object assigned as name of the MATLAB Simulink Block is allowed to be edited. If there is no such element in the appropriate ddx – file, a new ADD object with classification “OUTPUT” will be created.
adaptDDXFileName [optional]	0	Do not try to rename a loaded project – ddx – file. (default)
	1	If the underlying ddx – file contains an ADD Project, the ddx – file will be renamed to project name. Hint: Backup file will be created at first!

Hint: “objType” has to be one of the follow: ‘online’, ‘parameter’, ‘cDefine’, ‘axis’, ‘map’.

“objType” will set only if specified ADD Container does not contain an element of name “objName”.

```
[result, error_string, ADDobjName] = vitADDOffline(“NewObject”, “online”);
```

```
[result, error_string, ADDobjName] = vitADDOffline(“NewObject”);
```

```
[result, error_string, ADDobjName] = vitADDOffline(“”, “parameter”, 1);
```

Hint: Service “vitADDOffline” has an additional return value!

Beside the “result” and “error_string” value there is yet another value which returns a char array. In case of a non empty char array “objName” or “loadMode” – value “1” as service parameter value, the third value “ADDobjName” returns the ADD conform name of the given object.

In order to use “vitADDOffline” in the “Simulink Mode” (loadMode = 1) see also Appendix [\[3.1 Integrating ADD into MATLAB Simulink\]](#)

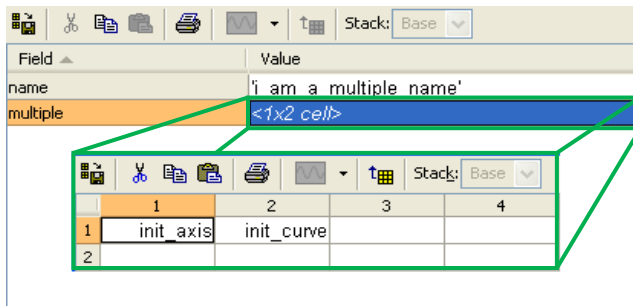
1.7 Service “vitCheckMultipleNames”

Service call: vitCheckMultipleNames(prefix, workspace);
 Alternative: vitCheckMultipleNames (prefix);

Service “vitCheckMultipleNames” checks the MATLAB WS variables specified by the parameter value “prefix” for multiple name occurrences in the embedded variable structures. Base Point is the structure field “adds_name”.

Parameter value	Value	Description
prefix [mandatory]	char array	Prefix of the MATLAB WS variables which will be processed.
workspace [optional]	char array	Specifies the MATLAB WS where the variables could be found. If there is no workspace declared MATLAB ‘base’ WS will be used instead.

[rv, result] = vitCheckMultipleNames(“prefix, workspace”);



On successful run (rv = 1) with no multiple names detected the **result** value will be an empty array.

Otherwise the **result** value will be a struct containing the detected multiple names (**result.names**) as well as a cell array (**result.multiple**) with the appropriate MATLAB WS variables.

If there occurs an error during the service run **rv** will be set to ‘0’. In this case the **result** value contains the appropriate error string.

1.8 Service “vitADDCreateNewContainer”

Service call: vitADDCreateNewContainer(CntName);
 Alternative: vitADDCreateNewContainer();

Service “vitADDCreateNewContainer” is designed to create an empty ddx – file. File will be stored at “adds.ddxfile” location.

It is also possible to label the newly created container with a user specific name assigned in parameter value “CntName” or “adds” variable “adds.add_environment”.

Parameter value	Value	Description
CntName [optional]	Empty char array	Create new ddx-file. Container name will be set to “adds.add_environment”. If “add_environment” is an empty char array or not assigned, container name will be labeled as “Empty 1.1.0”. (default)
	char array	Create new ddx-file. Container name will be set to “CntName”.

1.9 Service “vitDDXvsWS”

Service call: vitDDXvsWS(skipList, htmlPath);
 Alternative: vitDDXvsWS(skipList); vitDDXvsWS ();

Service “vitDDXvsWS” checks given MATLAB WS variables (Characteristic of the appropriate variables is the prefix assigned by [adds.prefix](#)) against a ddx – file which is specified in [adds.ddxfile](#).

Parameter value	Value	Description
skipList [optional]	Empty cell array	All elements will be processed. (default)
	cell array of strings	List of elements which will be skipped.
htmlPath [optional]	Empty char array	Do not display and save the result as an html - file.
	char array	Display and save the result as an html - file.

[retval, result, err_string] = vitDDXvsWS(skipList, htmlPath)

Beside the “result” and “error_string” value there is yet another value which returns an array of structs.

Field	Value
name	'Test'
element	'value'
ddx_content	'1'
ws_content	'0'

[result.name](#): Name of the dissimilar element.

[result.element](#): Name of appropriate struct field.

[result.ddx_content](#): Content of the ddx element.

[result.ws_content](#): Content of the ws element.

1.10 Service “vitAccessMetalInfos”

Service call: vitAccessMetalInfos(access, key, value);
 Alternative: vitAccessMetalInfos(access, key);

Service “vitAccessMetalInfos” writes/reads ddx-file ([adds.ddxfile](#)) related “meta” information.

Parameter value	Value	Description
access [mandatory]	char array	Access type! Either “set” (write access) or “get” (read access)
key [mandatory]	char array	Specifies the value which should be set or get.
value [optional]	char array	Specifies the key value on access type “set”

[retval, result] = vitAccessMetalInfos(“set”, “PRODUCER”, “Visu-IT!”)

[retval, result] = vitAccessMetalInfos(“get”, “PRODUCER”)

If there occurs an error during the service run [retval](#) will be set to ‘0’. In this case the [result](#) value contains the appropriate error string. On successful process run ([retval=1](#)), the get method will return the appropriate values of the selected key as [result](#).

1.11 Service “vitDAT2WS” (Trial Version – DCM Only!)

Service call: vitDAT2WS(Verbose, pValue, aInfo);

Alternative: vitDAT2WS(Verbose, pValue) vitDAT2WS(Verbose); vitDAT2WS();

Service “vitDAT2WS” writes initial data-file ([adds.initfile](#)) information to the current MATLAB WS Variables.

Parameter value	Value	Description
Verbose [optional]	0	Do not create an executable MATLAB m – file. (default)
	1	Create an executable MATLAB m – file representing the init-file content
pValue [optional]	0	Omit value setting on plane MATLAB WS Variables
	1	Set also plane MATLAB WS Variable values
aInfo [optional]	0	Do not set additional init-file information.
	1	Update MATLAB WS Variable values based on the additional init-file information (Object LongName, Unit)

`[retval, result] = vitDAT2WS("1", "1", "0")`

If there occurs an error during the service run `retval` will be set to '0'. In this case the `result` value contains the appropriate error string. On successful process run (`retval=1`), the get method will return the appropriate values of the selected key as `result`.

1.12 Service “vitWS2DAT” (Trial Version – DCM Only!)

Service call: vitWS2DAT(Verbose);

alternative: vitWS2DAT();

Service “vitWS2DAT” writes the current MATLAB WS Variable information to an initial data-file ([adds.initfile](#)).

Parameter value	Value	Description
Verbose [optional]	0	Do not create an executable MATLAB m – file. (default)
	1	Create an executable MATLAB m – file representing the init-file content

`[retval, result] = vitWS2DAT("1")`

If there occurs an error during the service run `retval` will be set to '0'. In this case the `result` value contains the appropriate error string. On successful process run (`retval=1`), the get method will return the appropriate values of the selected key as `result`.

2 Questions?

ADD Hotline

Tel.: +49 (0)941 – 49082-16
email: hotline@visu-it.com

ADD Product page

Internet: visu-it.de/products/add/

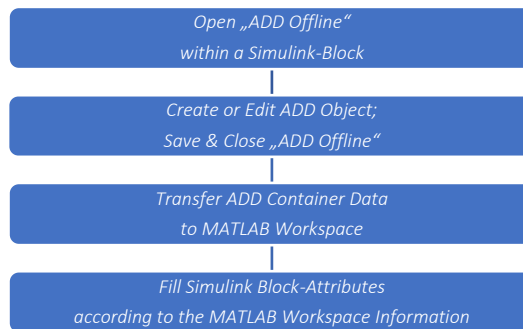
3 Appendix

3.1 Integrating ADD into MATLAB Simulink

Like all MATLAB functions it's possible to embed "ADD MATLAB Services" into your "MATLAB Simulink" or "dSpace TargetLink" work flow.

In order to parallel the creation of a simulation model and an appropriate "ADD" – Container or Project (allegorised as DDX-file) it's necessary to adduct several services.

Schematic diagram:



This example shows up the implementation for a Simulink "Lookup2D" - block!

The aim is to ally the creation of the "Lookup2D" – block and its corresponding ADD object ("Map") by using the services "vitADDOffline" and "vitDDX2WS".

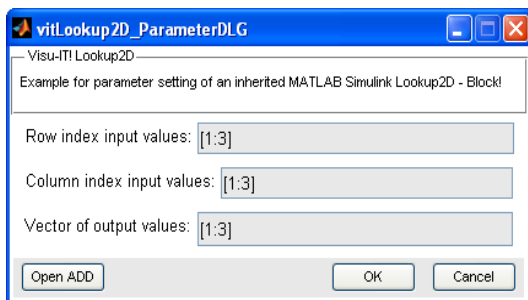
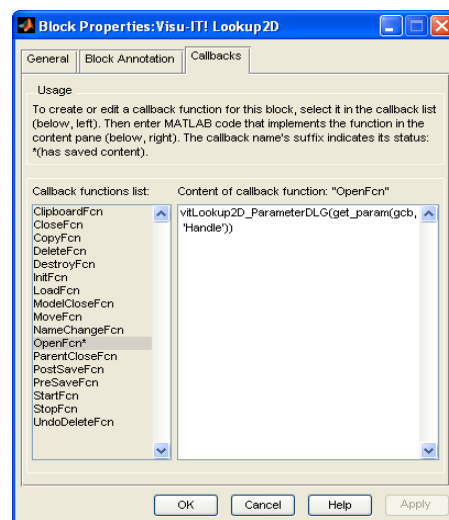
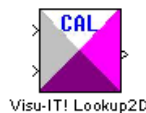
1. Create & Link an individual "Function Block Parameter" (q.v. "vitLookup2D_ParameterDLG.fig")

To link a figure on a block element it is necessary to call the decated MATLAB function within the appropriated Callback function.

The Callback function "OpenFcn" hab been used in this particular example.

After double clicking on the appropriate block, "OpenFcn" executes the dialog function "vitLookup2D_ParamterDLG.m".

The parameter value is defined as handle of the current block (MATLAB function: "get_param(gcf, 'Handle')").



The "vitLookup2D_ParameterDLG" block property window can handle the row, column and output data of a "Lookup2D" - block. Also it is possible to start "ADD Offline" by pressing the "Open ADD" button, set the block attributes ("RowIndex", "ColumnIndex", "OutputValues") by pressing "OK" and close the property window by pressing "Cancel".

2. Create ADD object using “ADDOffline”

By pressing the “Open ADD” button the underlying “btADD_Callback” function will be executed.

```

File Edit Text Desktop Window Help
110 % --- Executes on button press in btADD.
111 function btADD_Callback(hObject, eventdata, handles)
112 % hObject handle to btADD (see GCBO)
113 % eventdata reserved - to be defined in a future version of MATLAB
114 % handles structure with handles and user data (see GUIDATA)
115 try
116     global bhL2D; global addsPrefix; global ADDObjName;
117     hd_tbOut = get(handles.tbOut, 'String');
118     %                                     add object type
119     %                                     |
120     [result, err_string, ADDObjName] = vitADDOffline( hd_tbOut , 'map', 1);
121     %                                     |
122     %                                     set initial name of the new add object to block name run in "Simulink" mode
123     if(result==1 & ~isempty(ADDObjName))
124         %                                     transfer initial values
125         %                                     |
126         [retVal, retStr] = evalin('base', 'vitDDX2WS(0,0,1);');
127         %                                     |
128         %                                     do not create an additional m-script create plane workspace variables
129         if(retVal)
130             blType = get_param(bhL2D, 'BlockType');
131             MapFound = ~isempty(findstr(lower(blType), 'lookup2d'));
132             if(MapFound)
133                 addsPrefix = evalin('base', 'adds.prefix;');
134                 set(handles.tbRow, 'String', evalin('base', [addsPrefix, 'map.', ADDObjName, '.x_axis_name']));
135                 set(handles.tbCol, 'String', evalin('base', [addsPrefix, 'map.', ADDObjName, '.y_axis_name']));
136                 set(handles.tbOut, 'String', ADDObjName);
137             end
138         end
139     elseif(isempty(ADDObjName)) return; end; %Finished by User
140     if(~result || ~retVal) warndlg(err_string, 'Warning!'); end
141 catch err = lasterror; errorDlg(err.message, 'Error!', 'on'); end
    
```

Executing service “vitADDOffline” in Simulink mode

Executing service “vitDDX2WS”

Receiving MATLAB Workspace Info

First service call “vitADDOffline” opens “ADD Offline” user interface and automatically creates a new object with classification “Output”, type “Map” and set the “Vector of output values” – string as name. At this junction object name will be “[1:3]”. After redacting the newly created ADD object to a proper “Map”-object, the “ADD Offline” editor must save the DDX-file and could be closed afterward.

The subsequent service call “vitDDX2WS” transmits the DDX-file into MATLAB Workspace (WS).

Now it’s possible to exactly snatch the content of a WS variable by composing its prefix (see “addsPrefix”), type (according to the block type: Lookup2D » Map) and name (“ADDObjName”).

3. Assign Simulink Block

WS data could be transferred to Simulink block (identifier: “bhL2D”) by pressing the “OK”-bottom.

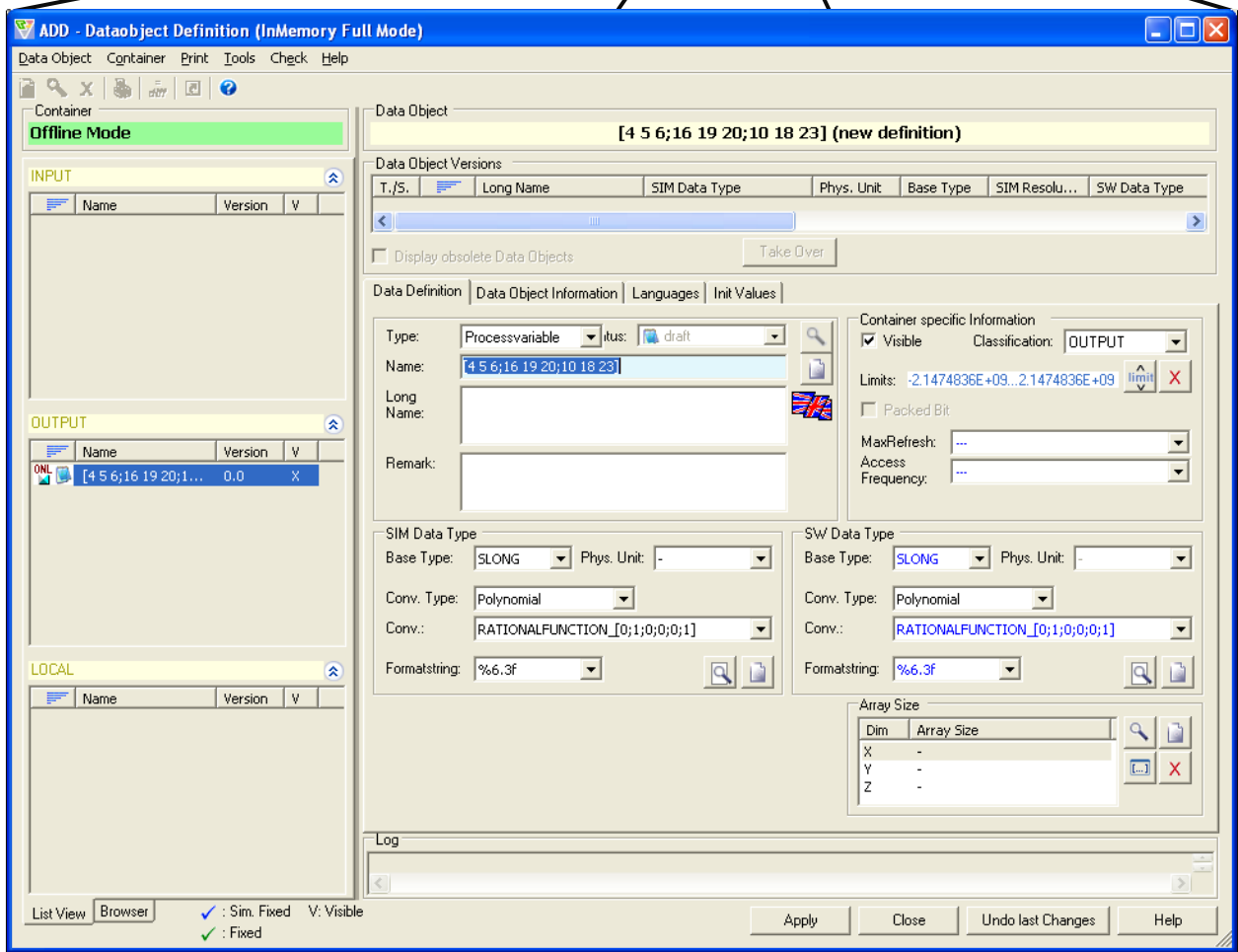
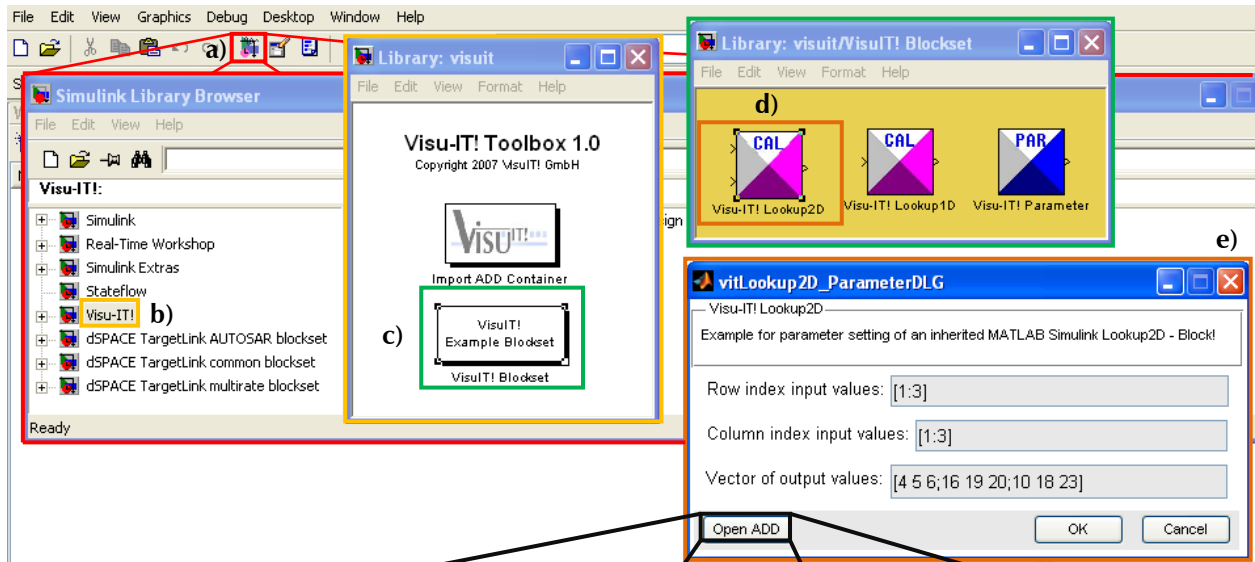
```

File Edit Text Desktop Window Help
94 % --- Executes on button press in btOkay.
95 function btOkay_Callback(hObject, eventdata, handles)
96 % hObject handle to btOkay (see GCBO)
97 % eventdata reserved - to be defined in a future version of MATLAB
98 % handles structure with handles and user data (see GUIDATA)
99 try
100     global bhL2D;
101
102     Co = get(handles.tbCol, 'String'); Ro = get(handles.tbRow, 'String'); Ou = get(handles.tbOut, 'String');
103     set_param(bhL2D, 'ColumnIndex', Co, 'RowIndex', Ro, 'OutputValues', Ou);
104     close;
105 catch
106     err = lasterror;
107     errorDlg(err.message, 'Error!', 'on');
108 end
    
```

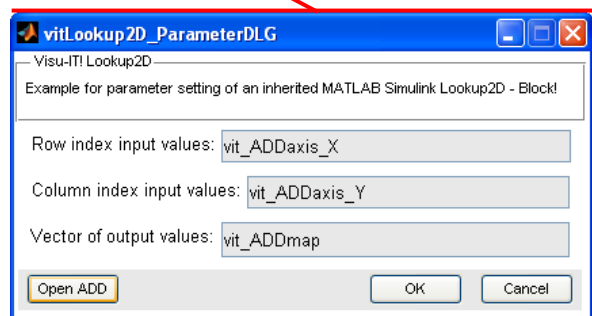
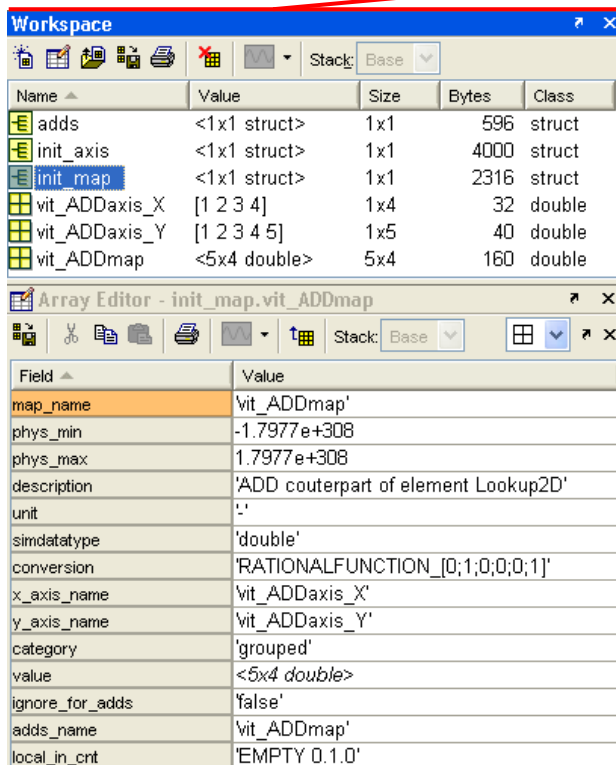
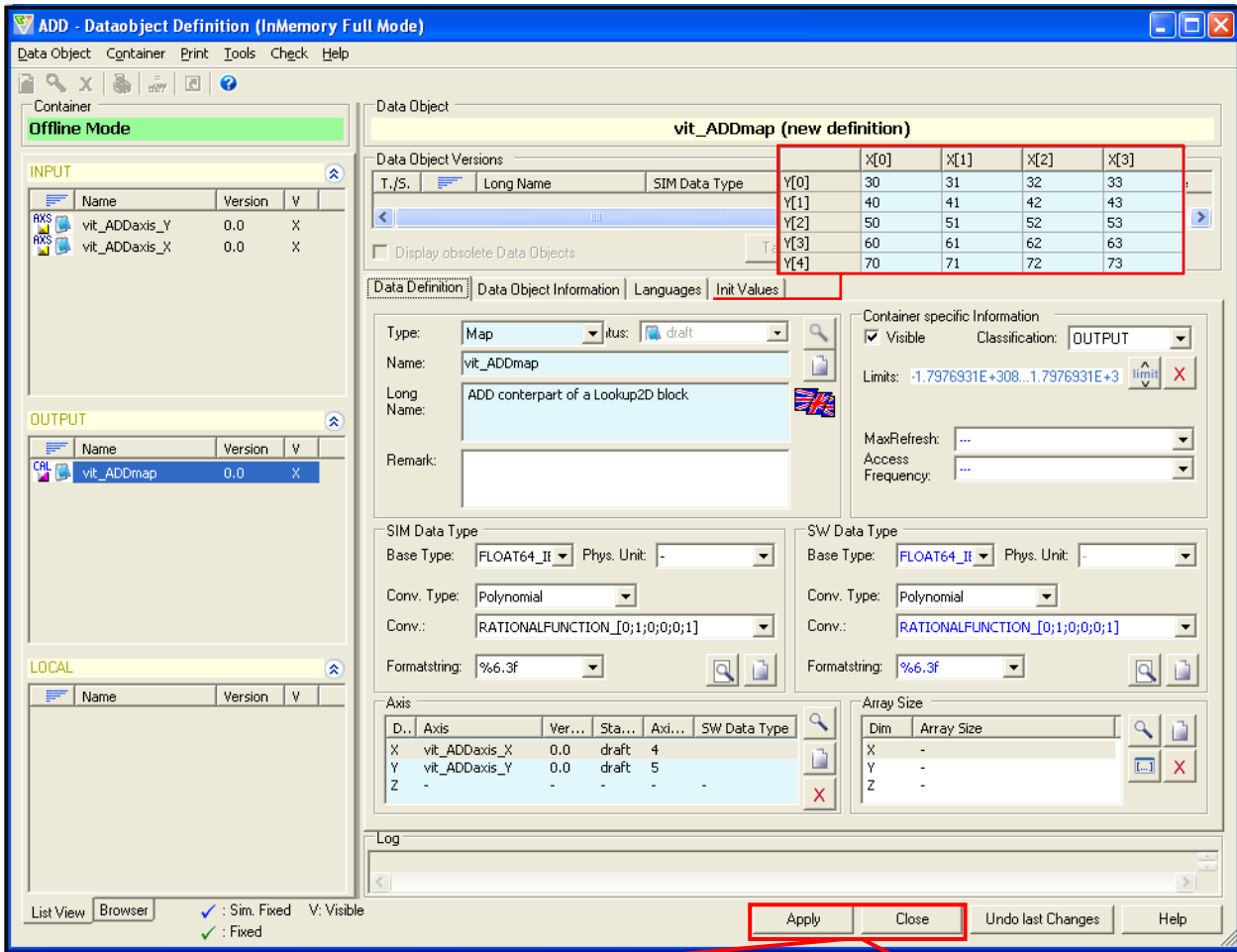
MATLAB Simulink function; set block parameter

Hint: Block parameters could only be set if the underlying model is unlocked!

3.1.1 Building History



g)

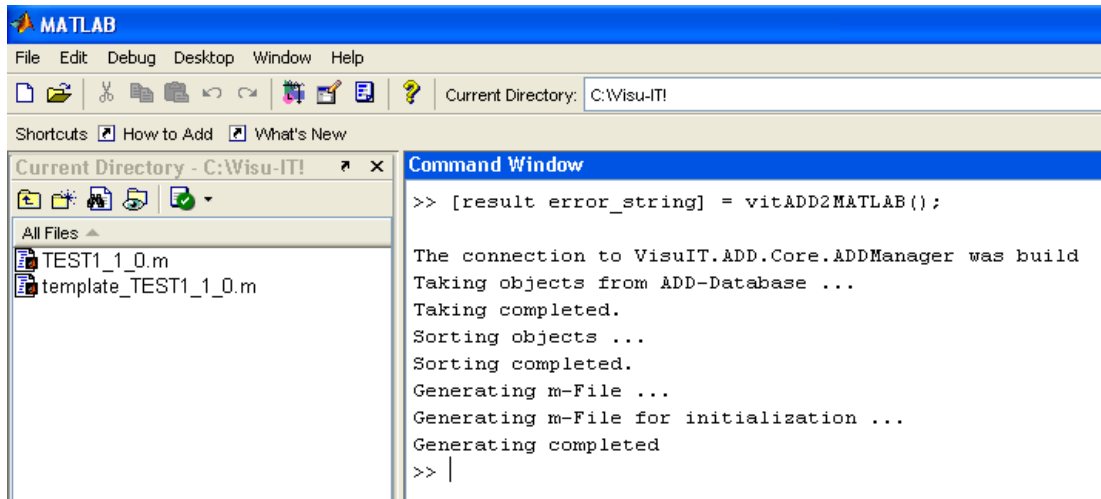


Legend:

- a) Open "Simulink Library Browser".
- b) Select "Visu-IT!" library.
- c) Open "Visu-IT! Blockset".
- d) Select a Block.
- e) Open parameter dialog.
- f) Open "ADD Offline" (Button "Open ADD").
- g) Adapt ADD element.
- h) Save and Close "ADD Offline".
- i) MATLAB Workspace & Parameter dialog will be filled automatically.

3.2 Legacy Import

1. Generate m-File only:



MATLAB Command Window: ADD import info; MATALB Current Directory: Generated m – scripts.

At one hand “SRS-Filename”.m – file (see: „TEST1_1_0.m“) declares variables for MATLAB WS and at the other hand “template_” – file declares the initial value arrays of the variables.

“SRS-Filename”.m – file clipping:

```

1  %-----
2  % File Info:
3  % File Name           : C:\Visu-IT!\TEST1_1_0.m
4  % Creation Time of File : 18-Jul-2007 17:16:35
5  % Container Info:
6  % Container           : TEST1.1.0
7  % Description         : test discription part1
8  %                    : part2
9  %                    : part3
10 % Version             : 1.0
11 % Status              : simulation fixed
12 % Creation Time      : 08.03.2006 11:29:31
13 % Created by         : addadmin
14 % Last Update        : 08.03.2006 11:29:32
15 % Updated by         : addadmin
16 %-----
17
18
19 % Description         : test discription part1
20 % Axis axis_test1
21 % Defined in         : TEST1.1.0
22 % Version            : 1.0
23 % Status             : fixed
24 % Creation Time     : 08.03.2006 11:31:08
25 % Created by        : addadmin
26 % Last Update       : 08.03.2006 16:59:21
27 % Updated by        : addadmin
28 - init_axis.axis_test1.axis_name = 'axis_test1';
29 - init_axis.axis_test1.phys_min = -2147483648;
30 - init_axis.axis_test1.phys_max = 2147483647;
31 - init_axis.axis_test1.description = 'first test axis; ';
32 - init_axis.axis_test1.unit = '-';
33 - init_axis.axis_test1.points = int32(5);
34 - init_axis.axis_test1.values = int32([0 0 0 0 0]);
35 - init_axis.axis_test1.simdatatype = 'int32';
36 % End Axis axis_test1
37

```

File and Container Information

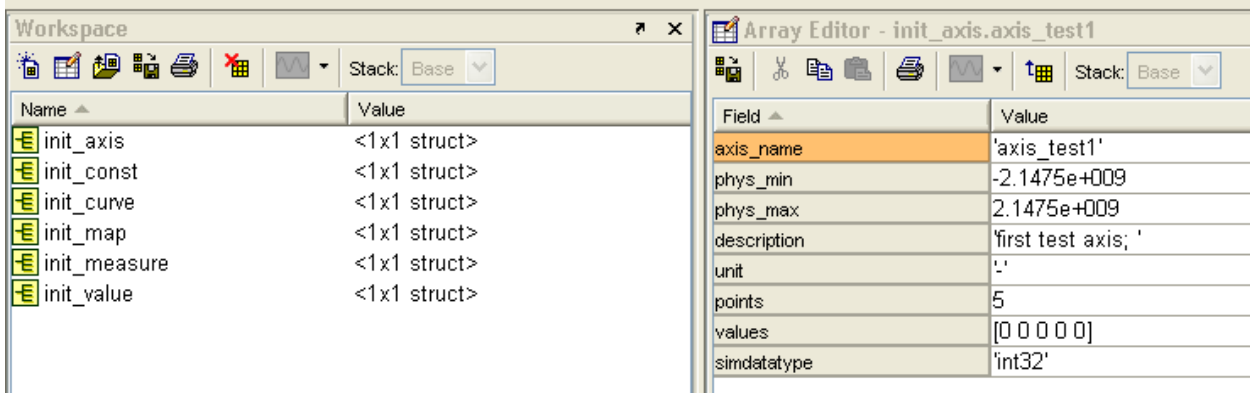
Container Object - Information

MATLAB variables

2. Import to MATLAB Workspace (incl. m-File!):

See also “1. Generate m – file only”!

Additional functionality of “Generate m – file only” which generates MATLAB Workspace variables.



Variable structure : <Prefix><Objekttyp>.<Objektattribute>

- **Prefix:** Prefix of the MATLAB WS variables.

Objekttyp:	MABLAB Denotation	<->	ADD Denotation
	value	<->	Parameter
	const	<->	cDefine
	axis	<->	Axis
	curve	<->	Map (1d, Kennlinie)
	map	<->	Map (2d, Kennfeld)
	measure	<->	Online

- **Objektattribute:**

Structure fields defined for all objects:

< Objekttyp>_name:	Object name
phys_min, phys_max:	Minimum and maximum of the physical values
description:	Object description
unit:	Physical unit
values:	Object values
simdatatype:	Object datatype

Additional structure field for object “axis”:

points:	Object value count
category:	Axis category

Additional structure field for object “curve”:

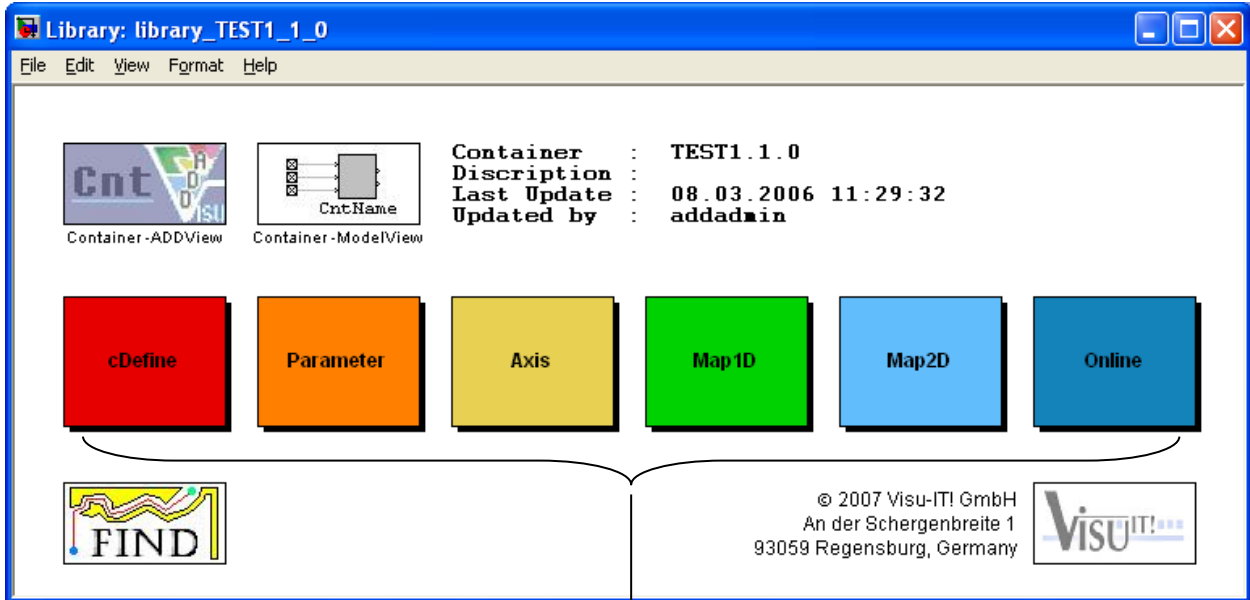
x_axis_name:	Name of the referenced axis
category:	Curve Category

Additional structure field for object “map”:

x_axis_name:	Name of the referenced x - axis
y_axis_name:	Name of the referenced y - axis
category:	Map Category

3. Create Container Library (incl. Container Model!):

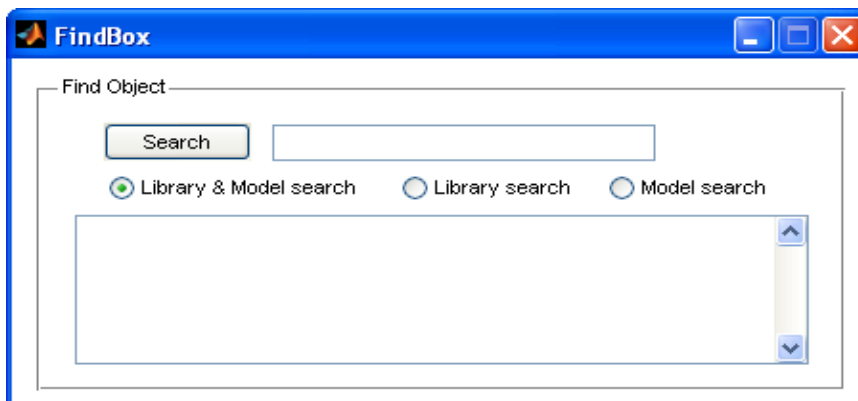
Creates a structured Simulink Library based on the ADD container objects and a MATLAB Workspace “struct” variable call “ADD” which contains all necessary information about the imported container. Each other imported ADD container will be added to the “ADD” structure or overridden if it already exists.



Note: The Container library could be handled as every other Simulink Library.

Container –ADDView: Opens container in ADD Offline Editor.

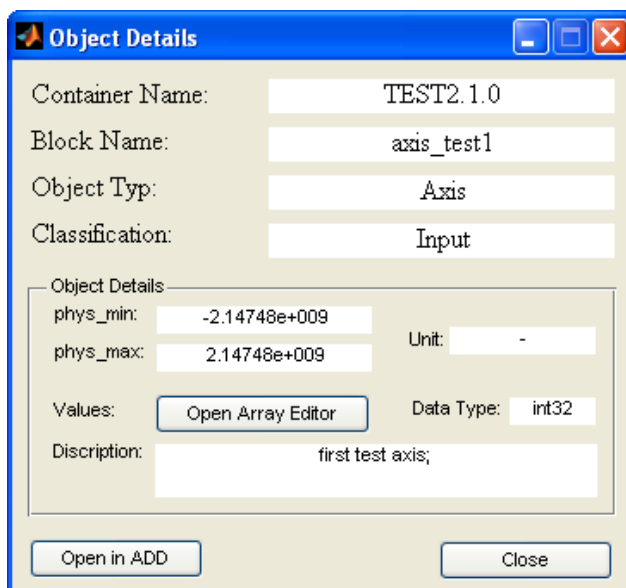
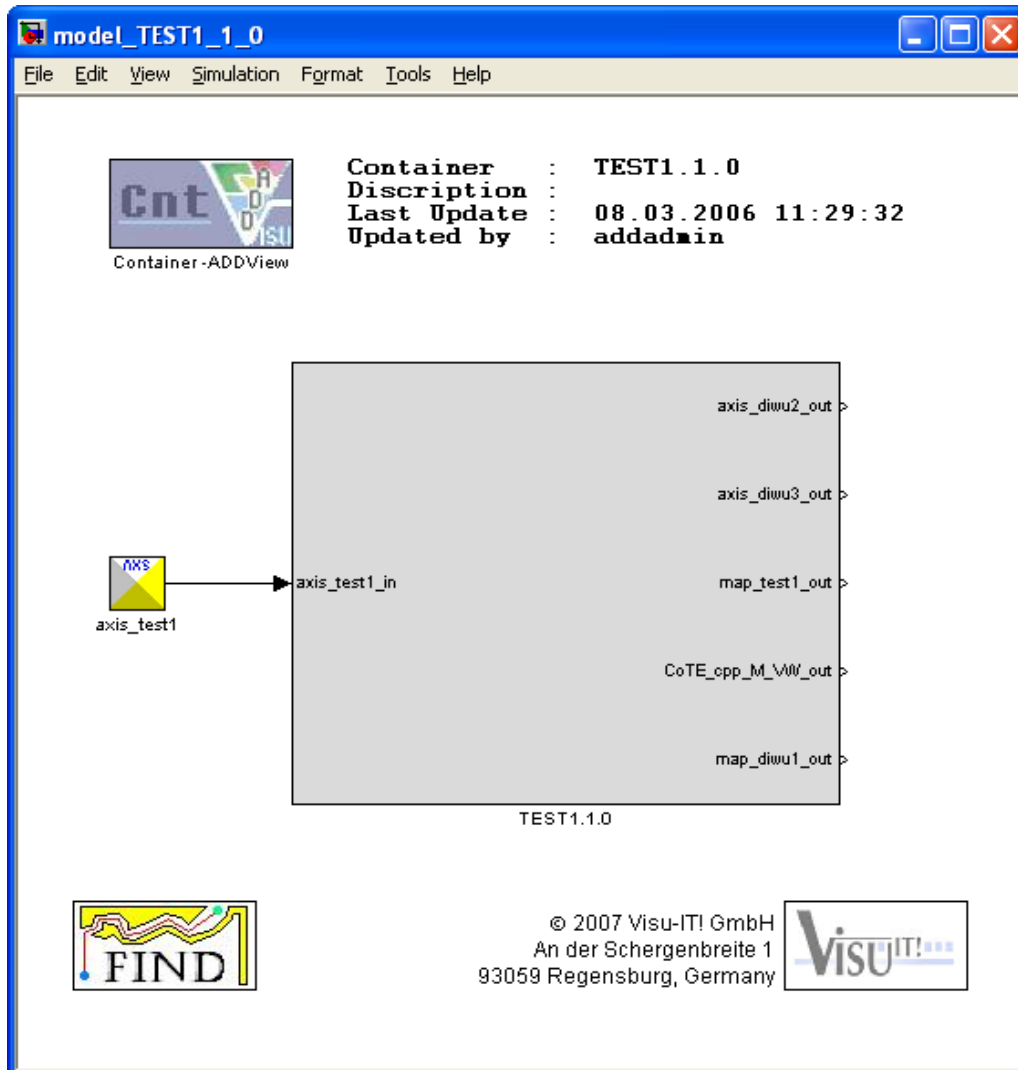
Find – Button: Opens the FindBox.



Help window for the search of container objects by name in the library and/or model.

By clicking on the appropriate list name Object will be shown and highlighted!

Container –ModelView: Opens a simplified model of the ADD container. The ADD container will be pictured as a Simulink Subsystem. Input objects of the container will be linked automatically with the container. Output and local objects will be placed inside the subsystem.



By double clicking on a container object the well arranged "Object Details" - window appears.

It shows content information of the appropriate Workspace variable.

The "Open Array Editor" – button displays the MATLAB Array Editor showing the appropriate values.

The "Open in ADD" – button grand a restricted access within the ADD Offline Editor. Only the object assigned as name of the MATLAB Simulink Block is allowed to be edited.

3.3 Structures used by MATLAB Services (Overview)

Global Struct „adds“		
Field	Datatype	Restriction
add_environment	STRING	Assigned by the service vitDDX2WS
ddxfile	STRING	-
mScript	STRING	-
pinfo (*)	SCALAR	'0', '1'
Prefix	STRING	-
verbose	SCALAR	'0', '1'

Element	Prefix+Structure-name	Type of the element in ADD
Axis	<xxx>_axis	axis
CodeVariant	<xxx>_codevariant	codevariant (data object & container attribute)
Constant	<xxx>_const	systemconstant
Conversion (*)	<xxx>_conversion	conversion definition
Curve	<xxx>_curve	map (1D)
Datatype (*)	<xxx>_datatype	data object data type definition (simulation & software)
Diagnose Index	<xxx>_diag_idx	diag_ddx
Map	<xxx>_map	map (2D)
Measurement	<xxx>_measure	online
Parameter	<xxx>_value	parameter
Structure	<xxx>_structure	structure

Field	Datatype	Restriction
<xxx>_name	STRING	-
task	STRING	-
adds_name	STRING	-
array_size_x	SCALAR[1x1]	>0
array_size_y	SCALAR[1x1]	>0
axis_type	STRING	'x_axis', 'x_y_axis', 'x_y_z_axis'
basetype (*)	STRING	'enum', 'single', 'double', 'bool', 'intX', 'uintX' (X= 8, 16, 32, 64)
bit_carrier	STRING	-
bit_position	SCALAR [1x1]	-
catalog (*)	STRING	-
category	STRING	'grouped', 'individual'
compresseddatatype	STRING	-
condition	STRING	-
conversion (*)	STRING	-
customtags	CELL-Array	{,tag', ,description', ,kind'}
dcm_displayname	STRING	-
dcm_formatstring	STRING	-
dcm_function	STRING	-

defined_in_cnt	CELL	-
description	STRING	-
displayname (*)	STRING	-
formatstring (*)	STRING	-
ignore_for_adds	SCALAR [1x1]	'0', '1'
is_array_x	SCALAR [1x1]	'0', '1'
is_array_y	SCALAR [1x1]	'0', '1'
is_enumeration	SCALAR [1x1]	'0', '1'
local_in_cnt	CELL	-
mem_sec	STRING	-
member	STRUCT	Element 'structure', 'axis', 'curve', 'map', 'parameter', 'measurement'
member_count	SCALAR [1x1]	-
member_list	CELL	-
member_list_substruct	CELL	-
member_name	STRING	-
member_type	STRING	-
maxaxispoints	SCALAR [1x1]	-
online_reference	STRING	-
phys_max	SCALAR [1x1]	-
phys_min	SCALAR [1x1]	-
record_type	STRING	-
record_type_memreg	STRING	'ram', 'rom'
reference_value	SCALAR	-
referenced_in_cnt	CELL	-
simdatatype (*)	STRING	-
size_x_from_system_constant	STRING	-
size_y_from_system_constant	STRING	-
status	STRING	-
swdatatype (*)	STRING	-
type (*)	STRING	'enum', 'rationalfunction', 'string', 'linear'
unit (*)	STRING	-
value	SCALAR	-
values (*)		enum: [{'string', scalar[1x1]}; ...] rationalfunction: [{scalar[1x1], ...}; ...] string: [{'string', scalar[1x1], scalar[1x1]}; ...] linear: [{scalar[1x1], scalar[1x1]}; ...]
visible	SCALAR [1x1]	'0', '1'
write_only	SCALAR [1x1]	'0', '1'
x_axis_name	STRING	-
y_axis_name	STRING	-

(*) The display of this field depends on the value of adds-struct "pinfo"!
 pinfo = 0: "basetype", "displayname", "swdatatype", "type" "values" not available!

Structure schedule:

Structure of onlines in "list measure":

<online_name>.measure_name
<online_name>.phys_min
<online_name>.phys_max
<online_name>.description
<online_name>.status
<online_name>.customtags
<online_name>.unit (*)
<online_name>.formatstring (*)
<online_name>.simdatatype (*)
<online_name>.swdatatype (*)
<online_name>.compresseddatatype
<online_name>.conversion (*)
<online_name>.is_enumeration
<online_name>.bit_carrier
<online_name>.bit_position
<online_name>.write_only
<online_name>.value
<online_name>.reference_value
<online_name>.visible
<online_name>.mem_sec
<online_name>.task
<online_name>.ignore_for_adds
<online_name>.adds_name
<online_name>.defined_in_cnt
<online_name>.referenced_in_cnt
<online_name>.local_in_cnt
<online_name>.dcm_function
<online_name>.dcm_displayname

Structure of parameters in "list value":

<parameter_name>.value_name
<parameter_name>.phys_min
<parameter_name>.phys_max
<parameter_name>.description
<parameter_name>.status
<parameter_name>.customtags
<parameter_name>.unit (*)
<parameter_name>.formatstring (*)
<parameter_name>.simdatatype (*)
<parameter_name>.swdatatype (*)
<parameter_name>.compresseddatatype
<parameter_name>.conversion (*)
<parameter_name>.is_enumeration
<parameter_name>.value
<parameter_name>.is_array_x
<parameter_name>.array_size_x
<parameter_name>.size_x_from_system_constant
<parameter_name>.is_array_y
<parameter_name>.array_size_y
<parameter_name>.size_y_from_system_constant
<parameter_name>.reference_value
<parameter_name>.visible
<parameter_name>.mem_sec
<parameter_name>.task
<parameter_name>.ignore_for_adds
<parameter_name>.adds_name
<parameter_name>.defined_in_cnt
<parameter_name>.referenced_in_cnt
<parameter_name>.local_in_cnt
<parameter_name>.dcm_function
<parameter_name>.dcm_displayname
<parameter_name>.dcm_formatstring

Structure of axis in "list_axis":

<axis_name>.axis_name
<axis_name>.phys_min
<axis_name>.phys_max
<axis_name>.description
<axis_name>.status
<axis_name>.customtags
<axis_name>.unit (*)
<axis_name>.formatstring (*)
<axis_name>.value
<axis_name>.reference_value
<axis_name>.visible
<axis_name>.mem_sec
<axis_name>.task
<axis_name>.simdatatype (*)
<axis_name>.swdatatype (*)
<axis_name>.compresseddatatype
<axis_name>.conversion (*)
<axis_name>.is_enumeration
<axis_name>.online_reference
<axis_name>.maxaxispoints
<axis_name>.category
<axis_name>.axis_type
<axis_name>.ignore_for_adds
<axis_name>.adds_name
<axis_name>.defined_in_cnt
<axis_name>.referenced_in_cnt
<axis_name>.local_in_cnt
<axis_name>.dcm_function
<axis_name>.dcm_displayname
<axis_name>.dcm_formatstring

Structure of constants in "list_const":

<constant_name>.const_name
<constant_name>.phys_min
<constant_name>.phys_max
<constant_name>.description
<constant_name>.status
<constant_name>.customtags
<constant_name>.unit (*)
<constant_name>.formatstring (*)
<constant_name>.value
<constant_name>.reference_value
<constant_name>.simdatatype (*)
<constant_name>.swdatatype (*)
<constant_name>.compresseddatatype
<constant_name>.conversion (*)
<constant_name>.is_enumeration
<constant_name>.ignore_for_adds
<constant_name>.adds_name
<constant_name>.defined_in_cnt
<constant_name>.referenced_in_cnt
<constant_name>.local_in_cnt

Structure of maps in "list_map":

<map_name>.map_name
<map_name>.phys_min
<map_name>.phys_max
<map_name>.description
<map_name>.status
<map_name>.customtags
<map_name>.unit (*)
<map_name>.formatstring (*)
<map_name>.simdatatype (*)
<map_name>.swdatatype (*)
<map_name>.compresseddatatype
<map_name>.conversion (*)
<map_name>.is_enumeration
<map_name>.x_axis_name
<map_name>.y_axis_name
<map_name>.value
<map_name>.reference_value
<map_name>.category
<map_name>.visible
<map_name>.mem_sec
<map_name>.task
<map_name>.ignore_for_adds
<map_name>.adds_name
<map_name>.defined_in_cnt
<map_name>.referenced_in_cnt
<map_name>.local_in_cnt
<map_name>.dcm_function
<map_name>.dcm_displayname
<map_name>.dcm_formatstring

Structure of onlines in "list_diag_idx":

<diag_idx_name>.measure_name
<diag_idx_name>.phys_min
<diag_idx_name>.phys_max
<diag_idx_name>.description
<diag_idx_name>.status
<diag_idx_name>.customtags
<diag_idx_name>.unit (*)
<diag_idx_name>.formatstring (*)
<diag_idx_name>.simdatatype (*)
<diag_idx_name>.swdatatype (*)
<diag_idx_name>.compresseddatatype
<diag_idx_name>.conversion (*)
<diag_idx_name>.is_enumeration
<diag_idx_name>.value
<diag_idx_name>.reference_value
<diag_idx_name>.visible
<diag_idx_name>.mem_sec
<diag_idx_name>.task
<diag_idx_name>.ignore_for_adds
<diag_idx_name>.adds_name
<diag_idx_name>.defined_in_cnt
<diag_idx_name>.referenced_in_cnt
<diag_idx_name>.local_in_cnt
<diag_idx_name>.dcm_function
<diag_idx_name>.dcm_displayname

Structure of curves in "list_curve":

<curve_name>.curve_name
<curve_name>.phys_min
<curve_name>.phys_max
<curve_name>.description
<curve_name>.status
<curve_name>.customtags
<curve_name>.unit (*)
<curve_name>.formatstring (*)
<curve_name>.simdatatype (*)
<curve_name>.swdatatype (*)
<curve_name>.compresseddatatype
<curve_name>.conversion (*)
<curve_name>.is_enumeration
<curve_name>.x_axis_name
<curve_name>.value
<curve_name>.reference_value
<curve_name>.category
<curve_name>.visible
<curve_name>.mem_sec
<curve_name>.task
<curve_name>.ignore_for_adds
<curve_name>.adds_name
<curve_name>.defined_in_cnt
<curve_name>.referenced_in_cnt
<curve_name>.local_in_cnt
<curve_name>.dcm_function
<curve_name>.dcm_displayname
<curve_name>.dcm_formatstring

Structure of structures in "list_structure":

<structure_name>.structure_name
<structure_name>.phys_min
<structure_name>.phys_max
<structure_name>.description
<structure_name>.status
<structure_name>.record_type
<structure_name>.record_type_memreg
<structure_name>.member_count
<structure_name>.member_list
<structure_name>.member_list_substruct
<structure_name>.member
<structure_name>.value
<structure_name>.reference_value
<structure_name>.category
<structure_name>.visible
<structure_name>.mem_sec
<structure_name>.task
<structure_name>.ignore_for_adds
<structure_name>.adds_name
<structure_name>.defined_in_cnt
<structure_name>.referenced_in_cnt
<structure_name>.local_in_cnt
<structure_name>.dcm_function
<structure_name>.dcm_displayname
<structure_name>.dcm_formatstring

Structure of codevariants in "list_codevariant":

<codevariant_name>.codevar_name
<codevariant_name>.condition
<codevariant_name>.description

Structure of datatypes in "list_datatype":

<datatype_name>.datatype_name
<datatype_name>.basetype
<datatype_name>.conversion
<datatype_name>.formatstring
<datatype_name>.unit
<datatype_name>.adds_name

Structure of conversions in "list_conversion":

<conversion_name>.conversion_name
<conversion_name>.displayname
<conversion_name>.type
<conversion_name>.values
<conversion_name>.catalog
<conversion_name>.adds_name

(*) The display of this field depends on the value of adds-struct "pinfo"! (pinfo = 0: "basetype", "displayname", "swdatatype", "type" "values" not available!)