Flux API

Contents

Available Languages

1

This paper describe C interface.

For other languages, please go to "%INSTALLFLUX%\Api"

Available languages:

- C : 64 bits Windows/Linux
- Java : 64 bits Windows/Linux
- Excel VBA : 64 bits Windows
- Flux Jython : 64 bits Windows/Linux
- Jython : 64 bits Windows/Linux
- Python : 64 bits Windows/Linux (use at leat Python 3.0, reference version : Python 3.5.4)
- Matlab : 64 bits Windows
- Scilab : 64 bits Windows
- Fortran : 64 bits Windows

flux_mp.dll, flux_mp.lib, flux_mp.h

2

The file "flux_mp.h" is located in "%INSTALLFLUX%\Api\.x.\include

The file "flux_mp.lib" is located in "%INSTALLFLUX%\Api\.x.\lib"

The file "flux_mp.dll" is located in "%INSTALLFLUX%\Bin\dll\.x.dll"

Where .x. is:

- “win64” for Windows 64 bits
- “lin64” for Linux 64 bits

Functions to manage Server

3

This chapter covers the following:

FMP_init

```
H_ERROR FMP_init (char* configFile,CEDINT32 debugFlag)
```

Initialize configuration, environment, localisation etc.

- Input:
  - configFile: configuration file to initialize "flux_mp.dll"

    Nothing : installed version
  - debugFile: debug flag.

    If debugFlag equal DEBUGMODE a python file "fmp_debug.py" is generated in current directory.
- Return:
  - error handle (NULL=OK)

Valid value for debugFlag in file header "flux_mp.h"
RELEASEMODE
DEBUGMODE

Exemple:

```
status = FMP_init("", RELEASEMODE)
```

FMP_arg

Convert argument

```
H_ERROR FMP_arg(char* argName,char* valArg , char** arg)
```

- Input:
  - argName: argument name
  - valArg: argument value
- Output:
  - arg: convert argument
- Return:
  - error handle (NULL=OK)

Predefined arguments

Name	Description	Default value
LANGUAGE_LABEL	Language: 1 # French 2 # English	2
CONSOLE_LABEL	Display the console: YES # Display	
GUI_MEMORY_LABEL	Java memory in MB	600
CHARACTER_MEMORY_LABEL	Characters memory in B	50000000
NUMERICAL_MEMORY_LABEL	Numerical memory in B	500000000

Example:

```
char** Args;
…
Args = malloc(2*sizeof(char*));
Args[0]=FMP_arg(NUMERICAL_MEMORY_LABEL,"600000000");
Args[1]=FMP_arg(LANGUAGE_LABEL,"2");
```

Other available arguments:

Name	Description	Default value
EXT_MACRO_DIR	Macro directory	${SERVER_INSTALL_DIR}/Extensions/Macros
EXT_OVERLAY_DIR	Overlays directory	${SERVER_INSTALL_DIR}/Overlays
FLUX_NCORES	Number of cores	
USER_CLASS_PATH	User class path	
USER_JYTHON_PATH	Jython user path	
USER_LIB_PATH	Library user path	
USER_PATH	User version path	

Example:

```
char** Args;
…
Args = malloc(2*sizeof(char*));
Args[0]=FMP_arg("FLUX_NCORES","4");
```

FMP_startLocaleServer

Initialize server

```
H_ERROR FMP_startLocaleServer (char* server, char* wrkDir,char** Args,CEDINT32
 nbArgs)
```

- Input:
  - Server: Server to start

    "FLUX3D_64" : start Flux3d 64 bits
  - wrkDir: Working directory
  - Args: Additional arguments
  - nbArgs: Number of additional arguments
- Return:
  - serverUid: Unique identifier for server
- Example: Start Flux3d in 64 bits version with 600 Mb of memory with English language

```
char** Args;
 …
Args = malloc(2*sizeof(char*));
Args[0]=FMP_arg(NUMERICAL_MEMORY_LABEL,"600000000");
Args[1]=FMP_arg(LANGUAGE_LABEL,"2");
serverUid = FMP_startLocaleServer(FLUX3D_64,LOCALEWRKDIR,Args,2);
```

FMP_stopServer

Stop and close server

```
H_ERROR FMP_stopServer (CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return:
  - error handle (NULL=OK)

FMP_killServer

Kill server

```
H_ERROR FMP_killServer (CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return:
  - error handle (NULL=OK)

Functions to manage Project

4

This chapter covers the following:

FMP_loadProject

Load project

```
H_ERROR FMP_loadProject (CEDINT32 serverUid,char* projectName)
```

- Input:
  - serverUid: server Uid
  - projectName: project name
- Return:
  - error handle (NULL=OK)

FMP_saveProject

Save project

```
H_ERROR FMP_saveProject (CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return: error handle (NULL=OK)

FMP_saveProjectAs

Save project as

```
H_ERROR FMP_saveProjectAs (CEDINT32 serverUid,char * projectName)
```

- Input:
  - serverUid: server Uid
  - projectName: project name
- Return:
  - error handle (NULL=OK)

FMP_closeProject

Close project without save

```
H_ERROR FMP_closeProject(CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return:
  - error handle (NULL=OK)

Functions to manage VariationParameter

5

This chapter covers the following:

FMP_getAllVariationParameter

Get names of all Input/Output parameter (variation parameter)

```
H_ERROR FMP_getAllVariationParameter(CEDINT32 serverUid, CEDINT32* nbrParam,char***
 paramName)
```

- Input:
  - serverUid: server Uid
- Output:
  - nbrParam: number of I/O parameter
  - paramName:Array of parameter names (free with "FMP_freeChar")
- Return:
  - error handle (NULL=OK)

FMP_getVariationParameterProperties

Get the number of real and the number of component for a list of VariationParameter

```
H_ERROR FMP_getVariationParameterProperties (CEDINT32 serverUid,CEDINT32
 nbrParam,char** paramName ,CEDINT32* nbrReal,CEDINT32* nbrComponents,CEDINT32*
 typeParam)
```

- Input:
  - serverUid: server Uid
  - nbrParam: number of I/O parameter (variation parameters)
  - paramName: name of I/O parameter
- Output:
  - nbrReal: array of number of real (free with " FMP_free")
  - nbrComponent: array of number of components (free with " FMP_free")
  - typeParam: array of parameter type (free with " FMP_free")
- Return:
  - error handle (NULL=OK)

Valid value for typeParam in file header "flux_mp.h"
UNKNOWNVARPARAM
UPDATABLEVARPARAM
NOUPDATABLEVARPARAM

FMP_setVariationParameterValue

Set VariationParameter value

```
H_ERROR FMP_setVariationParameterValue (CEDINT32 serverUid,char* paramName,double
 paramValue)
```

- Input:
  - serverUid: server Uid
  - paramName: name of I/O parameter
  - paramValue: value of I/O parameter
- Return:
  - error handle (NULL=OK)

FMP_getVariationParameterValue

Get VariationParameter value

```
H_ERROR FMP_getVariationParameterValue (CEDINT32 serverUid,char* paramName ,
 CEDINT32* nbrReal,CEDINT32* nbrComponents,double** valueParam)
```

- Input:
  - serverUid: server Uid
  - paramName: name of I/O parameter
- Output:
  - nbrReal: array of number of real (free with " FMP_free")
  - nbrComponents: array of number of components (free with " FMP_free")
  - valueParam:array of values (free with " FMP_free")
- Return:
  - error handle (NULL=OK)

Functions to manage SpatialParameter

6

This chapter covers the following:

FMP_getAllSpatialParameter

Get names of all Spatial Parameters

```
H_ERROR FMP_getAllSpatialParameter(CEDINT32 serverUid, CEDINT32* nbrParam,char***
 paramName)
```

- Input:
  - serverUid: server Uid
- Output:
  - nbrParam: number of Spatial Parameter
  - paramName: Array of parameter names (free with "FMP_freeChar")
- Return:
  - error handle (NULL=OK)

FMP_getSpatialParameterProperties

Get the properties of a list of Spatial parameters

```
H_ERROR FMP_getSpatialParameterProperties (CEDINT32 serverUid,CEDINT32
 nbrParam,char** paramName, CEDINT32* nbrReal,CEDINT32* nbrComponents,CEDINT32*
 typeParam)
```

- Input:
  - serverUid: server Uid
  - nbrParam: number of spatial parameters
  - paramName: Array of parameter names
- Output:
  - nbrReal: array of number of real (free with " FMP_free")
  - nbrComponents: array of number of components (free with " FMP_free")
  - typeParam: array of type (free with " FMP_free")
- Return:
  - error handle (NULL=OK)

Valid value for "typeParam" in file header "flux_mp.h"
UNKNOWNSPAPARAM
UPDATABLESPAPARAM
NOUPDATABLESPAPARAM

FMP_UpdateSpatialQuantityByImportRegion

Update Spatial Quantity with nodal values on region

```
H_ERROR FMP_UpdateSpatialQuantityByImportRegion (CEDINT32 serverUid, char*
 paramName,char* regionName,CEDINT32 dimRegion, CEDINT32 nbrNodes,CEDINT32*
 idNodes,CEDINT32 nbrReal,CEDINT32 nbrComponents,double* nodalValues )
```

- Input:
  - serverUid: server Uid
  - paramName: parameter name
  - regionName: region name
  - dimRegion: dimension region
  - nbrNodes: number of nodes
  - idNodes: Array of nodes
  - nbrReal: number of real by nodal value
  - nbrComponents: number of components by nodal value
  - nodalValues: array of nodal values
- Return:
  - error handle (NULL=OK)

FMP_UpdateSpatialQuantityByImportRegionByCoordinat

Update Spatial Quantity with nodal values on region

```
H_ERROR FMP_UpdateSpatialQuantityByImportRegionByCoordinates (CEDINT32 serverUid,
  char* paramName,char* regionName,CEDINT32 dimRegion, CEDDOUBLE localization,
 CEDINT32 mechanicalSetPosition, CEDINT32 nbrNodes,double* cooNodes,CEDINT32
 nbrReal,CEDINT32 nbrComponents,double* nodalValues )
```

- Input:
  - serverUid: server Uid
  - paramName: parameter name
  - regionName: region name
  - dimRegion: dimension region
  - localization: localization of coordinates
  - mechanicalSetPosition: mechanical set position
  - nbrNodes: number of nodes
  - cooNodes: Array of coordinates of nodes
  - nbrReal: number of real by nodal value
  - nbrComponents: number of components by nodal value
  - nodalValues: array of nodal values
- Return:
  - error handle (NULL=OK)

Valid value for " localization " in file header "flux_mp.h"
LOCALIZATIONNODETONODE
Localization value in meter

Valid value for " mechanicalSetPosition" in file header "flux_mp.h"
MECHANICALSETPOSITIONREFERENCE
MECHANICALSETPOSITIONCURRENT

FMP_getSpatialParameterPrecision

Return the precision of a spatial parameter

```
H_ERROR FMP_getSpatialParameterPrecision (char* paramName,double* precision)
```

- Input:
  - serverUid: server Uid
  - paramName: Array of parameter names
- Output:
  - precision: array of number of real (free with " FMP_free")
- Return:
  - error handle (NULL=OK)

Functions to manage MultiPointSupport

7

This chapter covers the following:

FMP_getAllMultiPointSupport

Get names of all MultiPointSupport

```
H_ERROR FMP_getAllMultiPointSupport(CEDINT32 serverUid, CEDINT32* CEDINT32*
 nbrMPS,char*** MPSName)
```

- Input:
  - serverUid: server Uid
- Output:
  - nbrMPS: number of MultiPointSupport
  - MPSName: Array of MultiPointSupport names (free with "FMP_freeChar")
- Return:
  - error handle (NULL=OK)

FMP_createMultiPointSupport

Create a multipoint support with a list of coordinates

```
H_ERROR FMP_createMultiPointSupport (CEDINT32 serverUid,char* supportName, CEDINT32
 nbrPoint,CEDDOUBLE* coordinatesPoint)
```

- Input:
  - serverUid: server Uid
  - supportName: MultiPointSupport name
  - nbrPoint: number of points
  - coordinatesPoint: coordinates of points
- Return:
  - error handle (NULL=OK)

FMP_getMultiPointSupportValuesWithDefaultRegion

Get values of a spatial formula on a multiPointsupport

```
H_ERROR FMP_getMultiPointSupportValuesWithDefaultRegion (CEDINT32 serverUid,
 char * supportName, char* spatialFormula, CEDINT32 regionDimension,char*
 defaultRegionName , CEDINT32* nbrValues,CEDINT32* nbrReal,CEDINT32*
 nbrComponents,double** supportValues)
```

- Input:
  - serverUid: server Uid
  - supportName: MultiPointSupport name
  - spatialFormula: spatial formula to compute
  - regionDimension: region dimension
  - defaultRegionName: default region name
- Output:
  - nbrValues: number of values
  - nbrReal: number of reals by value
  - nbrComponents: number of components by value
  - supportValues: Array of values (free with FMP_free)
- Return:
  - error handle (NULL=OK)

Functions to manage Regions

8

This chapter covers the following:

FMP_getAllRegion

Get names of all regions

```
H_ERROR FMP_getAllRegion (CEDINT32 serverUid, CEDINT32 dimRegion, CEDINT32*
 nbrRegion,char*** regionName)
```

- Input:
  - serverUid: server Uid
  - dimRegion: dimension region
- Output:
  - nbrRegion: number of region
  - regionName: Array of region names (free with "FMP_freeChar")
- Return:
  - error handle (NULL=OK)

FMP_getSpatialValuesAtNodesOnElementsOnRegion

Get values of spatial formula at nodes on elements on region

```
H_ERROR CEDCALL FMP_getSpatialValuesAtNodesOnElementsOnRegion (CEDINT32
 serverUid, char* spatialFormula,CEDINT32 dimRegion,char* regionName ,
 CEDINT32* nbrValues,CEDINT32* nbrReal,CEDINT32* nbrComponents, CEDINT32*
 nbrElements,CEDINT32** idElements, CEDINT32** nbrValuesOnElements,CEDDOUBLE**
 valuesAtNodesOnElements)
```

- Input:
  - serverUid: server Uid
  - spatialFormula: spatial formula
  - dimRegion: dimension region
  - regionName: region name
- Output:
  - nbrValues: number of values
  - nbrReal: number of reals by value
  - nbrComponents: number of components by value
  - nbrElements: number of elements
  - idElements: id of elements
  - nbrValuesOnElements: array of number of values by elements
  - valuesAtNodesOnElements: array of values at nodes on elements
- Return:
  - error handle (NULL=OK)

FMP_getSpatialValuesAtNodesOnRegion

Get values of spatial formula at nodes on region

```
H_ERROR CEDCALL FMP_ getSpatialValuesAtNodesOnRegion (CEDINT32 serverUid, char*
 spatialFormula,CEDINT32 dimRegion,char* regionName , CEDINT32* nbrValues,CEDINT32*
 nbrReal,CEDINT32* nbrComponents, CEDDOUBLE ** coordinatesAtNodes,CEDDOUBLE**
 valuesAtNodes)
```

- Input:
  - serverUid: server Uid
  - spatialFormula: spatial formula
  - dimRegion: dimension region
  - regionName: region name
- Output:
  - nbrValues: number of values
  - nbrReal: number of reals by value
  - nbrComponents: number of components by value
  - coordinatesAtNodes: array of coordinates of nodes
  - valuesAtNodesOnElements: array of values at nodes
- Return:
  - error handle (NULL=OK)

Functions to manage solver

9

This chapter covers the following:

FMP_solve

Solve completely the project

```
H_ERROR FMP_solve (CEDINT32 serverUid,char* scenarioName,char* projectName)
```

- Input:
  - serverUid: server Uid
  - scenarioName: scenario name
  - projectName: project name
- Return:
  - error handle (NULL=OK)

FMP_openMultiPhysicsSession

Open MultiPhysics session

```
H_ERROR FMP_openMultiPhysicsSession (CEDINT32 serverUid,char* scenarioName,char*
 projectName)
```

- Input:
  - serverUid: server Uid
  - scenarioName: scenario name
  - projectName: project name
- Return:
  - error handle (NULL=OK)

FMP_closeMultiPhysicsSession

Close MultiPhysics session

```
H_ERROR FMP_closeMultiPhysicsSession (CEDINT32 serverUid,char* scenarioName)
```

- Input:
  - serverUid: server Uid
  - scenarioName: scenario name
- Return:
  - error handle (NULL=OK)

FMP_solveActiveNextStep

Active next solver step

```
H_ERROR FMP_solveActiveNextStep (CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return:
  - error handle (NULL=OK)

FMP_solveCurrentStep

solve the current step

```
H_ERROR FMP_solveCurrentStep (CEDINT32 serverUid)
```

- Input:
  - serverUid: server Uid
- Return:
  - error handle (NULL=OK)

FMP_solveDefineNextStep

Define next step

```
H_ERROR FMP_solveDefineNextStep (CEDINT32 serverUid,CEDDOUBLE valueNextStep)
```

- Input:
  - serverUid: server Uid
  - valueNextStep: value next step
- Return:
  - error handle (NULL=OK)

Functions to manage Mesh

10

This chapter covers the following:

FMP_getMeshElementsOnRegion

Get mesh elements on region

```
H_ERROR FMP_getMeshElementsOnRegion (CEDINT32 serverUid, CEDINT32 dimRegion,char*
 nameRegion , CEDINT32* nbrElements,CEDINT32** idElements,CEDINT32**
 typeElements,CEDINT32** nbNodesByElements,CEDINT32** idNodesByElements)
```

- Input:
  - serverUid: server Uid
  - dimRegion: dimension region (2 or 3D)
  - nameRegion: name region
- Output:
  - nbrElements: number of elements
  - idElements: Array of element ids (free with "FMP_free")
  - typeElements: Array of element types (free with "FMP_free")
  - nbNodesByElements: Array of number of nodes by element (free with "FMP_free")
  - idNodesByElements: Array of id of nodes by elements (free with "FMP_free")
- Return:
  - error handle (NULL=OK)

FMP_getMeshElementsOnDomain

Get mesh elements on domain

```
H_ERROR FMP_getMeshElementsOnDomain (CEDINT32 serverUid, CEDINT32 dimDomain ,
 CEDINT32* nbrElements,CEDINT32** idElements,CEDINT32** typeElements, CEDINT32**
 nbNodesByElements,CEDINT32** idNodesByElements)
```

- Input:
  - serverUid: server Uid
  - dimDomain: dimension domain (2 or 3D)
- Output:
  - nbrElements: number of elements
  - idElements: Array of element ids (free with "FMP_free")
  - typeElements: Array of element types (free with "FMP_free")
  - nbNodesByElements: Array of number of nodes by element (free with "FMP_free")
  - idNodesByElements: Array of id of nodes by elements (free with "FMP_free")
- Return:
  - error handle (NULL=OK)

FMP_getMeshNodesOnRegion

Get mesh nodes on region

```
H_ERROR FMP_getMeshNodesOnRegion (CEDINT32 serverUid,CEDINT32 dimRegion,char*
 nameRegion , CEDINT32* nbrNodes,CEDINT32** idNodes)
```

- Input:
  - serverUid: server Uid
  - dimRegion: dimension region (2 or 3D)
  - nameRegion: name region
- Output:
  - nbrNodes: number of nodes
  - idNodes: Array of node ids(free with "FMP_free")
- Return:
  - error handle (NULL=OK)

FMP_getMeshNodesCoordinates

Get nodes coordinates

```
H_ERROR FMP_getMeshNodesCoordinates (CEDINT32 serverUid,CEDINT32 nbrNodes,CEDINT32*
 idNodes , CEDDOUBLE* coordinatesNodes)
```

- Input:
  - serverUid: server Uid
  - nbrNodes: number of nodes
  - idNodes: Array of node ids
- Output:
  - coordinatesNodes: array of node coordinates (dimension 3* nbrNodes)
- Return:
  - error handle (NULL=OK)

Functions to manage Jython command

11

This chapter covers the following:

FMP_executePythonCommand

Execute Jython command line

```
H_ERROR FMP_executePythonCommand (CEDINT32 serverUid,char* commandline)
```

- Input:
  - serverUid: server Uid
  - commandline: jython command line
- Return:
  - error handle (NULL=OK)

FMP_getJythonDoubleArrayValue

Get values of double Jython variable in an array

```
H_ERROR FMP_getJythonDoubleArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
CEDINT32* numberValues,CEDDOUBLE** values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
- Output:
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

FMP_getJythonIntArrayValue

Get values of integer Jython variable in an array

```
H_ERROR FMP_getJythonIntArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
 CEDINT32* numberValues,CEDINT32** values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
- Output:
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

FMP_getJythonStringArrayValue

Get values of string Jython variable in an array

```
H_ERROR FMP_getJythonStringArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
 CEDINT32* numberValues,TCHAR*** values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
- Output:
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

FMP_setJythonDoubleArrayValue

Set values of double Jython variable in an array

```
H_ERROR FMP_setJythonDoubleArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
 CEDINT32 numberValues,CEDDOUBLE* values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

FMP_setJythonIntArrayValue

Set values of integer Jython variable in an array

```
H_ERROR FMP_setJythonIntArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
 CEDINT32 numberValues,CEDINT32* values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

FMP_setJythonStringArrayValue

Set values of string Jython variable in an array

```
H_ERROR FMP_setJythonStringArrayValue (CEDINT32 serverUid,TCHAR* jythonVarName,
 CEDINT32 numberValues,TCHAR** values)
```

- Input:
  - serverUid: server Uid
  - jythonVarName: jython variable name
  - numberValues: number of values
  - values: array of values
- Return:
  - error handle (NULL=OK)

Functions to manage Memory

12

This chapter covers the following:

FMP_free

Free memory of array (int* or double*)

```
H_ERROR FMP_free (CEDINT32 serverUid,void** array)
```

- Input:
  - serverUid: server Uid
  - array: array to free
- Return:
  - error handle (NULL=OK)

FMP_freeChar

free memory of char array

```
H_ERROR FMP_free (CEDINT32 serverUid,void** array)
```

- Input:
  - serverUid: server Uid
  - array: array to free
- Return:
  - error handle (NULL=OK)

Functions to manage Error

13

This chapter covers the following:

FMP_getErrorMessage

Get error Message

```
const char * const FMP_getErrorMessage (H_ERROR errorId)
```

- Input:
  - errorId: error handle
- Return:
  - Error message or NULL if "errorId" is invalid

FMP_freeError

Free error handle

```
H_ERROR FMP_freeError (H_ERROR errorId)
```

- Input:
  - errorId: error handle
- Return:
  - error handle (NULL=OK)