EasternGraphics
visualize your business

Specification

GO IV (MP) - OFML Metaplanning – Concepts and Tables

Author	*Ekkehard Beier, Andreas Handschuh, EasternGraphics*	
Reference	Title	**Metaplanning Specification**
	Version	**1.17.0**
	Release Date	**2014-08-22**
	Implementation	**::ofml::go::1.17.3**
	Version of the document	**(0)**
History	At the end of the document	
State	Release to EGR, sales partners and customers. Subject to change due to technical progress, purpose of optimization and removal of conceptual errors.	
Feedback	metaplanning-core@EasternGraphics.com	
Remark	Changes relative to **MP 1.16.0(-0**) are marked by underline.	

NOTE: The implementation of this specification by a given run-time environment such as pCon.planner, pCon.xcad may be incomplete!

Table of contents

1 Introduction

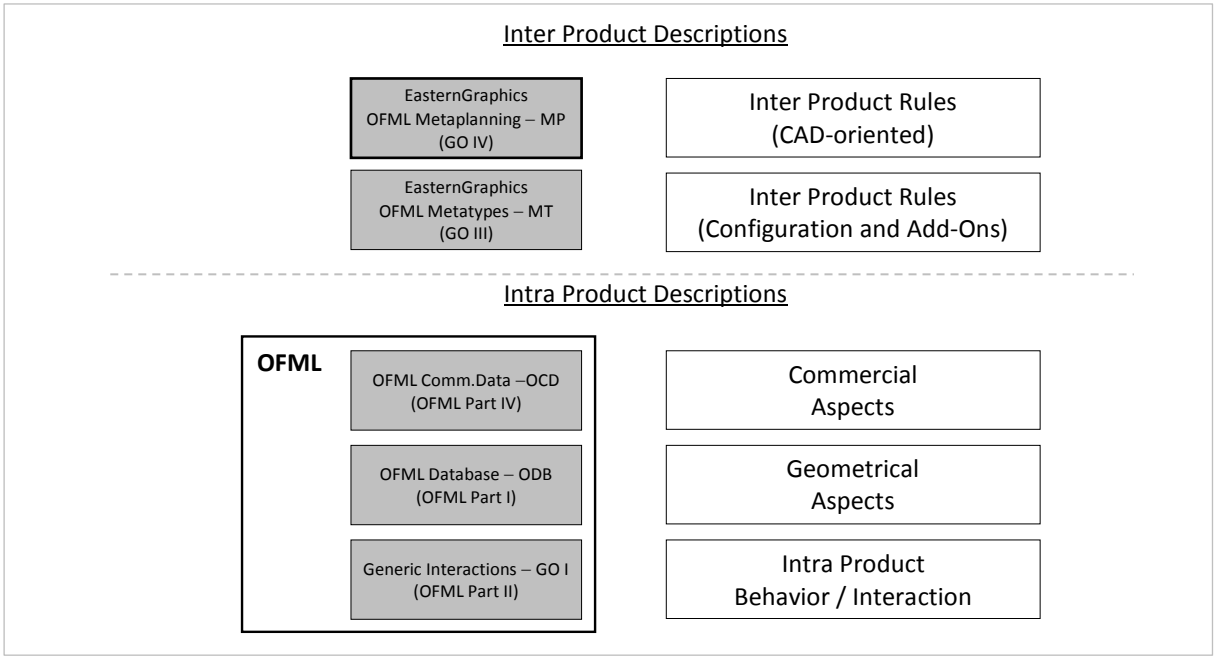
The concept of OFML Metaplanning adds the missing link from current OFML product models to the interactive design functionality of CAD systems. As Metatypes focus on

- Inter Product Configuration
- Inter Product Concatenation Rules and
- Inter Product Attachment Behavior

the scope of Metaplanning is

- Inter Product CAD-oriented Design Rules.

Metaplanning is designed to work as an additional layer above the Metatypes, but can also operate on native OFML articles.



The basic concept of Metaplanning is workflows. A workflow is complete interactive and visual article creation process. The result of a (successful) workflow is a number of articles.

Metaplanning provides a number of explicit workflows that are enumerated and described in the following section.

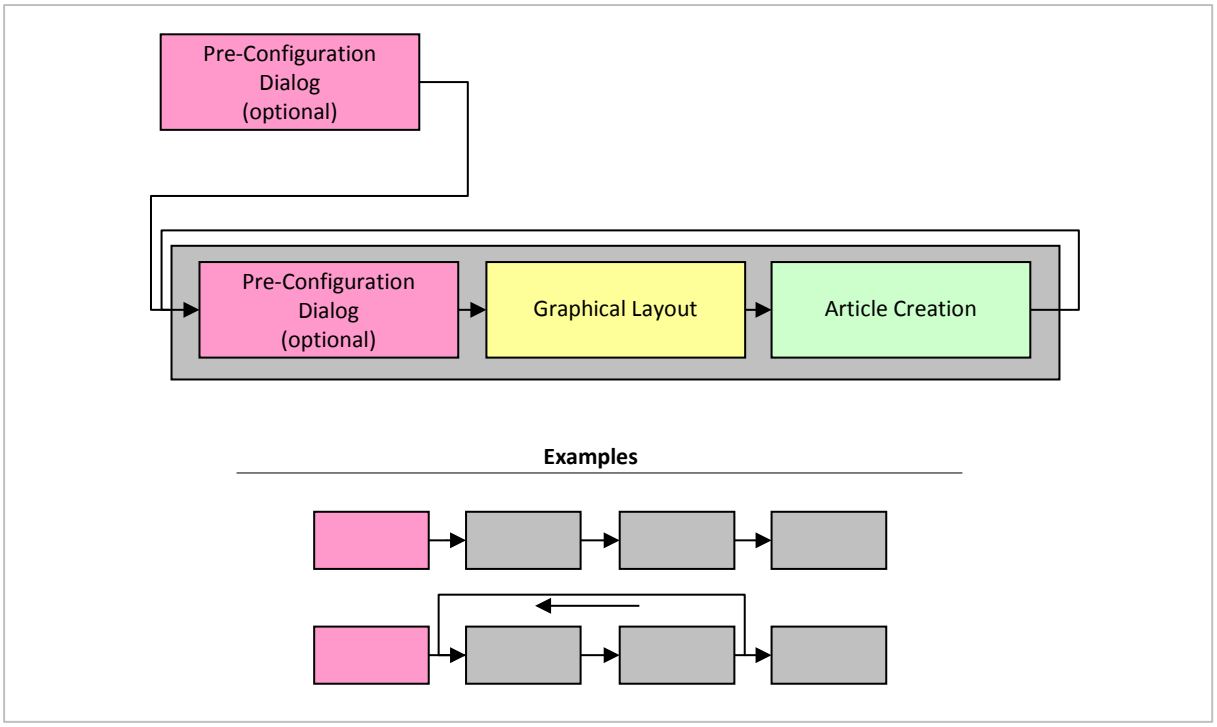
Each workflow implements a specific number of User Exits[1]. User Exits are implemented either in OFML or by any native environment. OFML-based user exits are per definition available on any Metaplanning implementation and therefore should be favored over native user exits. The class *::ofml::go::GoMetaPlanning* provides abstract user exits as well as utility methods for the implementation of OFML-based user exits. Any OFML-based user exit class must be derived (directly or indirectly) from the base class mentioned above.

2 Workflows

Metaplanning provides a number of explicit workflows that are enumerated and described in the following. The specific user exists are listed in the context of the related workflow. The workflow-independent user exits are documented at the end of this section.

2.1 Composite

The Composite workflow is an abstract workflow consisting of a number of non-abstract, basic workflows, which will be processed in the defined sequence.



User Exits

- **getNextChildWf() –** Returns the ID of the next child workflow. This way, a dynamic control of the sequence of child workflows is possible.

[1] See class *::ofml::go::GoMetaPlanning* for a detailed specification of the user exits including parameters and return value(s).

2.2 Selection

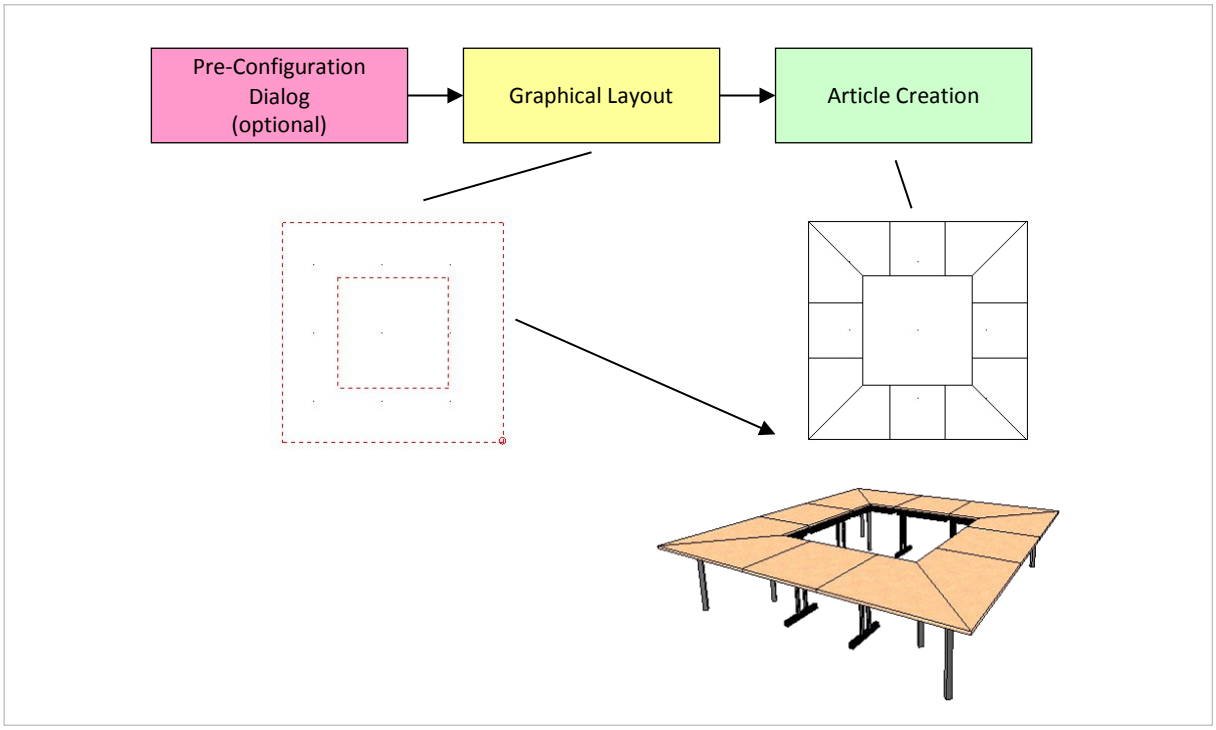
The Selection workflow is a basic workflow to be used as part of *Composite* workflows and supports the selection of one shape resp. article.

2.3 Position

The Position workflow is a basic workflow to be used as part of *Composite* workflows and supports the selection of the position of one shape resp. article. The shape itself cannot be selected but is predefined in the table.

2.4 Auto Fill

The Auto Fill workflow implements the creation of a number of articles from a single shape. The shape is either a basic geometrical object such as a rectangle, or a user-defined parametric geometry. The article creation is controlled either by CSV-based fill rule, or by an OFML user exit.

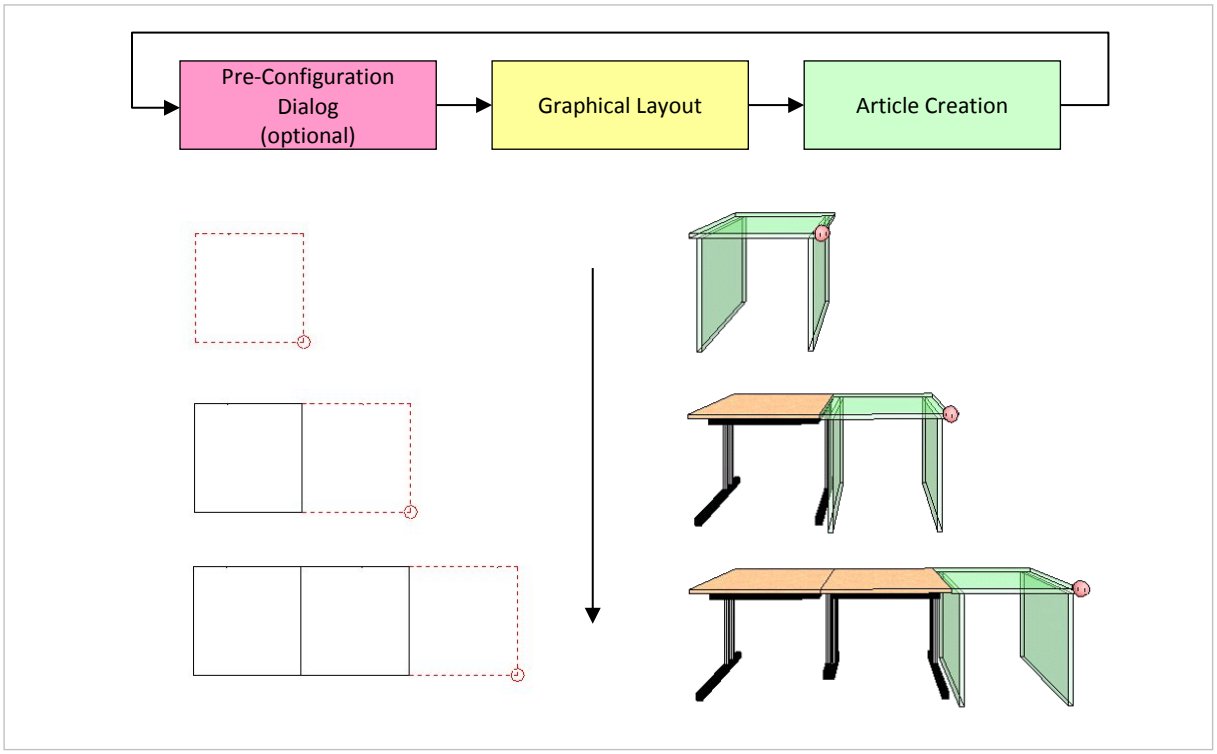


User Exits

- **createArticlesAF() –** Creates the articles as result of the workflow. If present, a possibly existing fill rule with the same ID will be ignored. It returns the position and orientation of the following workflow.

2.5 Repeated Insertion

The Repeated Insertion workflow implements the rule-based creation of a sequence of articles, and consists of an undefined number of steps. During each step a shape can be selected. The set of available shapes during each step as well as the default shape for the next step can be determined by user exits.



User Exits

- **getInitShapeRI()** – Returns the initial shape for a specific shape set by selecting one from the set of all possible shapes.
- **getNextShapeRI() –** Returns the next initial shape after the finalization of the previous insertion step. This can be used to set a context-sensitive default value. If not implemented (for a given workflow or context) the default shape will be detected by the Metaplanning implementation. If NULL is returned, the whole workflow will terminate.
- **getAvailableShapesRI() –** Returns a context-sensitive subset of the current shape set. This can be used to implement type-oriented limitations for a sequence of objects.
- **createArticlesRI() –** Creates the articles as result of the workflow. If the return value is not NULL, the Metaplanning tables for article creation will be ignored.

2.6 Resize

The Resize workflow allows an interactive resizing of an existing article. It is triggered by an Application Interactor (type *Resize*)[2].

User Exits

- **getObjDimensions()** – Returns the dimensions of the passed object ([Width, Height, Depth]).
- **setObjDimensions()** – Sets the dimensions of an object ([Width, Height, Depth]).
- **getShapeArgs()** – Returns the parameters of a shape (overrides mp_shapes (*args*)).

3 General User Exits

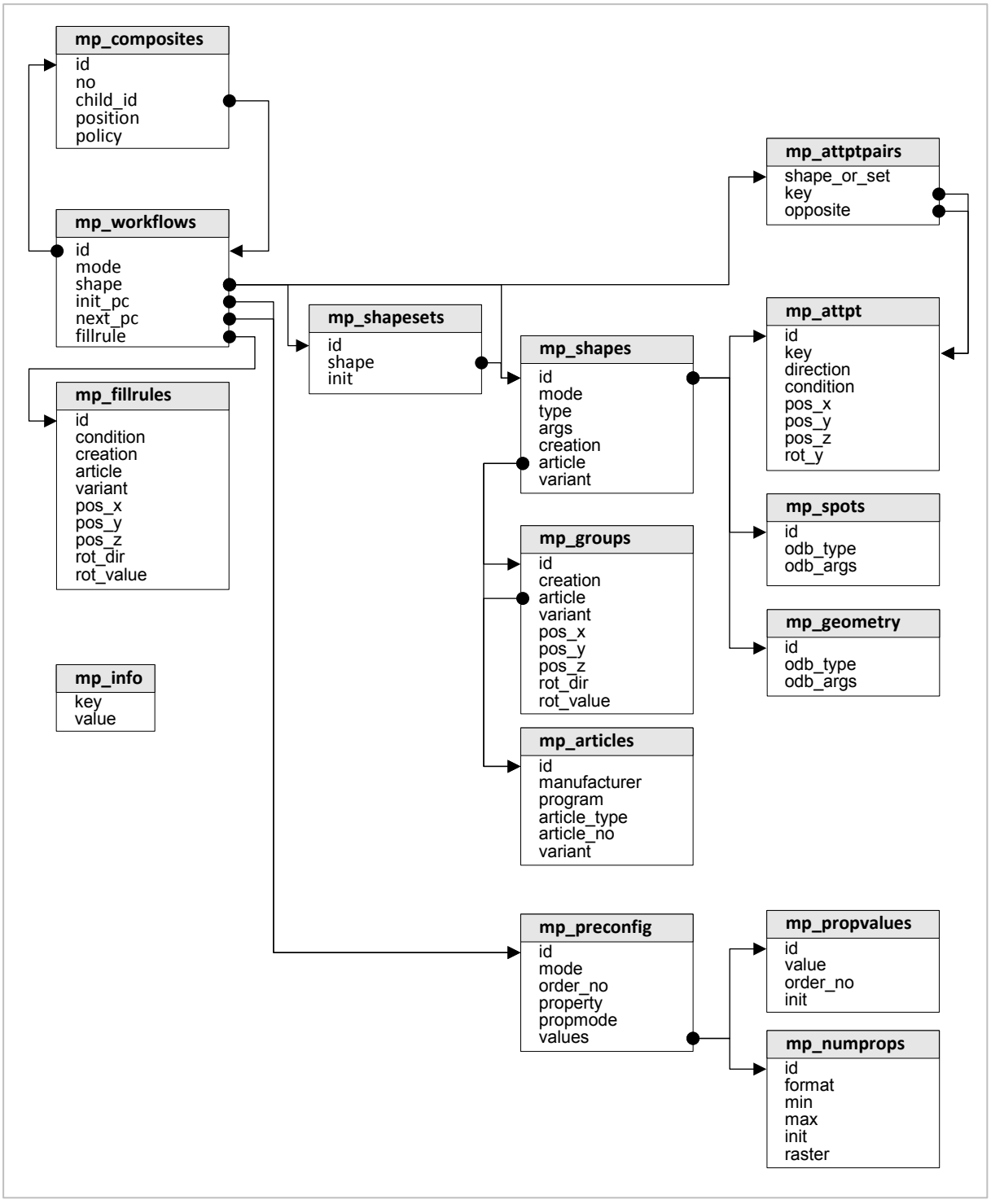
In the following the general User Exits that are available for any workflow, are defined. For a detailed description of parameters and return values see class *::ofml::go::GoMetaPlanning*.

- **setupPreConfiguration()** – Creates a user-customized pre-configuration dialog. As the potential functionality of OFML user exits is limited in this field, obviously native user exits are used to implement advanced kinds of pre-configuration.
- **customizePreConfiguration()** – Limit the values and ranges of standard pre-configuration properties.
- **finishWorkflow()** – To be called after the termination of a workflow. A parameter specifies if the workflow was finished regularly or canceled by the user, or was terminated by an error.
- **terminateWorkflow()** – Calling this User Exit terminates a workflow. Parameters specify if the workflow has to be terminated regularly or canceled and if the child workflow or the whole composite workflow has to be terminated.

[2] See Application Note AN-2013-001_Application_Interactors

4 Tables

In this chapter the MP tables will be described. For physical format see `mp.inp_descr`. The logical format (CSV tables) must be compiled into EBase representation, otherwise it won't work.



Global Enumerators

The following global enumerators are defined and used in one or more of the subsequently following tables.

Creation Mode
This mode controls what kind of article should be created as result of a workflow (step). Available creation modes are:

• metatype	A real OFML article, typically a Metatype object, should be created. In case of a Metatype, the actions will be triggered. For a native article the associated Metatype (if any) will be loaded.
• article	A real OFML article, typically a native article, should be created.
• item	A Basket item should be created.
• group	A group of articles should be created. The articles are described in the table *mp_groups*.

mp_workflows

The table *mp_workflows* is the main entry point for the whole MP. It sets up a number of workflows that are specified by the following entries and further tables.

Workflow ID (id)	This key defines a workflow and is used as primary key for the whole MP data base. Workflows must not start with an underscore (_). **Example**: *wf1*
Workflow Mode (mode)	This entry specifies the mode for the workflow and must be one of the following: • *composite* – See Workflow *Composite*. • *selection* – See Workflow *Selection*. • *position* – See Workflow *Position*. • *auto-fill* – See Workflow *Auto Fill*. • *repeated-insertion* – See Workflow *Repeated Insertion*. • *resize* – See Workflow *Resize*. **Example**: *auto-fill*
Shape (shape)	This entry links to a geometrical representation that is used as interactive visual feedback for the workflow. It either links to an entry in table *mp_shapesets* or *mp_shapes*. Workflows *Auto Fill* and *Resize* do only support shapes, not shape sets. Workflows *Repeated Insertion, Selection* and *Position* do only support shape sets, not single shapes. Workflow *Composite* does not support any shapes at all. **Example**: *sh1*

ID for Initial Pre-Configuration (init_pc)	If set, this key links to a pre-configuration dialog to be used at the beginning of workflows, defined in table *mp_preconfig*. Also for pre-configuration dialogs implemented by user exits, the key is used for identification. In this case, however, there is no entry in table *mp_preconfig*. The workflow *Resize* does not support any pre-configuration dialogs. **Example**: *pc1*
ID for Serial Pre-Configuration ID (next_pc)	If set, this key links to a pre-configuration dialog to be used between two workflow steps, defined in table *mp_preconfig*. Also for pre-configuration dialogs implemented by user exits, the key is used for identification. In this case, however, there is no entry in table *mp_preconfig.* If *next_pc* is empty, the property values of *init_pc* will be used as pre-configuration. In case of the Workflow *Auto Fill*, *next_pc* specifies a configuration dialog to be used after the shape has been drawn. The workflow *Resize* does not support any pre-configuration dialogs. **Example**: *pc1*
Fillrule ID (fillrule)	If set, this key links to a fill rule defined in table *mp_fillrules*. For fill rules implemented by user exits, this key is not used and there is no entry in table *mp_fillrules.* Only supported by workflow *Auto Fill*. **Example**: *fr1*

mp_composites

The table *mp_composites* defines a *Composite* workflow by a sequence of basic workflows. The termination state of the *Composite* workflow is that of the last processed child workflow.

Workflow ID (id)	This key refers to a workflow from table *mp_workflows* with mode composite. **Example**: *wf1*
Number (no)	This entry specifies the position of the child workflow in the sequence of all child workflows. It must be a non-negative, strictly monotonic increasing integer value. The child workflows will be processed then from small to bigger values. **Example**: *1*
Child Workflow ID (child_id)	This key refers to a workflow from table *mp_workflows* with any mode but not composite. **Example**: *wf1a*
Positioning (position)	The entry specifies a positioning policy for the workflow, and must be one of the following:

	• *interactive* – The workflow's position and orientation will be determined by user interaction. • *RS_R* – The last valid RS_R attach point and the _RS_R attach point of the initial shape of the new workflow will be used to determine position and orientation. • *rotation* – The workflow's orientation will be determined by user interaction. The position will be determined by using the last valid RS_R attach point and the _RS_R attach point of the initial shape of the new workflow. Note. For the first sub workflow and in case this workflow is not called again by *goto_<No>* (see below), this entry can be empty. For this initial calling of the first sub workflow this entry will be ignored in any case. **Example**: *RS_R*
Termination Policy (policy)	The entry specifies a termination policy and must be one of the following: • *standard* – No matter in what way the child workflow was terminated, the next child workflow will be activated. • *repeat* – If the child workflow was finished correctly, it should be repeated again. Otherwise the next child workflow is activated. • *terminate* – If the child workflow was terminated by cancellation, the whole composite workflow is terminated, i.e. cancelled. • *goto_<No>* - If the child workflow was finished correctly, go back to the workflow with number <No>. **Example**: *standard*, *goto_1*

mp_preconfig

The table *mp_preconfig* specifies user interface dialog controls, used for the pre-configuration.

Pre-Configuration ID (id)	This key defines a pre-configuration dialog. If multiple lines exist for one and the same ID, the dialog contains more than one property to choose. **Example**: *pc1*
Configuration Mode (mode)	This entry specifies the configuration mode of the dialog and must be one of the following: • *serial* – Serial configuration starting with the 1st property. After the 1st property was chosen, the next one is available for choice, and so on. This mode is not implemented yet. • *parallel* – Parallel configuration, i.e. at any time every property can be changed. • *serialCP* – Same as *serial*, but use previous settings of identical properties if available. This mode is not implemented yet. • *parallelCP* – Same as *parallel*, but use previous settings of identical properties if available. • *custom* – Customized configuration dialog[3]

[3] see *Module Spec GO Metadialog (GO VI)*

	In case of multiple properties, the mode of the 1st property is considered. Note. Obviously, for single-value pre-configuration dialogs this mode is irrelevant and therefore can be left empty. **Example**: *parallel*
Order Number (order_no)	In case of multiple properties this entry gives the order number of the property defined in this row. Order numbers begin with 1 and are incremented by 1 for each property. The order number is relevant: • to identify the 1st property in order to determine *mode* for the dialog • to get the display sequence of the properties • to get the configuration sequence if *mode* is *serial* **Example**: *1*
Property (property)	The entry names the property that has to be configured. It's the name of the Metatype or native property that has to be set then. The property can also be evaluated in user exits. In this case, the name can be chosen according to the OFML language rules. **Example**: *GProgram*
Property Mode (propmode)	The mode defines the kind of the property and can be: • *choice* – A standard choice list property. • *numeric* – An integer or float input property. **Example**: *choice*
Values ID (values)	This key links to a set of further property values defined in tables *mp_propvalues* (mode *choice*) or *mp_numprops* (mode *numeric*). **Example**: *program1*

mp_propvalues

The table *mp_propvalues* supports property values to be used by the pre-configuration dialogs.

Property ID (id)	This key defines a property value set to be referenced from table *mp_preconfig*. **Example**: *program1*
Property Value (value)	This key specifies a property value. **Example**: *MF1*
Order Number (order_no)	This entry gives the order number of the property value defined in this row. Order numbers begin with 1 and are incremented by 1 for each property value. The order number is relevant to get the display sequence of the property values. **Example**: *1*

Initial Value (init)	The initially selected value is marked with 1. Otherwise 0 is used. If there is no explicitly marked initial property value then it is determined by the Metaplanning implementation in undefined manner. **Example**: *1*

mp_numprops

The table *mp_numprops* implements numeric property values to be used by the pre-configuration dialogs.

Please make sure that

- The initial value is in the range defined by minimum and maximum.
- The maximum is greater than the minimum.
- If defined, the raster contains the minimum, the maximum and the initial value.

If any of these rules is hurt, the behavior of the MP run-time environment is formally undefined and may not correspond to your expectations.

Note. In case of a Length property the measurement unit is Meter. In case of an Arc property the measurement unit is Radiant.

Property ID (id)	This key defines specific parameters for a numeric property from table *mp_preconfig*. **Example**: *width*
Property Format (format)	This key specifies the property format and must be one of the following: • *@I* – A universal integer property. • *@F* – A universal float property. • *@L* – A specific float property representing a length value. • *@A* – A specific float property representing an arc value. **Example**: *@F*
Minimum Value (min)	This entry gives the minimum value for the range, which is a valid value of that range. **Example**: *1.0*
Maximum Value (max)	This entry gives the maximum value for the range, which is a valid value of that range. **Example**: *5.0*
Initial Value (init)	This entry sets an initial value for the property. **Example**: *2.0*
Raster Value (raster)	This entry defines a raster to be applied on the input of values. A valid raster must be a positive number. Use 0 (or 0.0) to disable rasterization explicitly. **Example**: *1.0*

mp_shapesets

The table *mp_shapesets* implements a set of shapes. A shape set is only needed for the *Repeated Insertion*, *Selection* and *Position* workflows. One and the same shape can be contained in different shape sets. Shape sets are not allowed to be elements of shape sets, again.

Shape Set ID (id)	This key defines a shape set. **Example**: *ss1*
Shape ID (shape)	This entry links to a shape defined in table *mp_shapes*. **Example**: *sh1*
Initial Shape (init)	The initial shape is marked with 1. Otherwise 0 is used. One and only one initial shape must exist for any shape set. **Example**: *1*

mp_shapes

The table *mp_shapes* defines shapes to be used either inside a shape set, or directly by a workflow.

Shape ID (id)	This key defines a shape. Multiple lines with the same ID are illegal. **Example**: *sh1*
Shape Mode (mode)	The mode controls the interactive behavior of the shape. Available modes are: • *free* – There is no restriction for the interactive modification of the shape. • *lx-lz* – First select local x, then local z coordinate. • *lz-lx* – First select local z, then local x coordinate. • *gx-gz* – First select global x, then local z coordinate. • *gz-gx*– First select global z, then local x coordinate. Note. Use mode *free* when modification is restricted by the parameters (see below) to 1 axis. The shape mode is only needed in the workflows *Auto Fill* and *Resize*. **Example**: *free*
Shape Type (type)	The type controls the visual feedback of the shape. Available types are: • *dyn-line* – A dynamic line. • *fix-line* – A fix line that can be moved and rotated, but not resized. • *geometry* – A user-defined geometry. In the Workflow *Auto Fill* scaling of the geometry is given by ODB parameters *W* (width) and *D* (depth). • The related geometry will be linked by table *mp_geometry*, however, the ODB arguments in that table will be ignored. In this case the Shape ID is also used as primary key (id) for the table *mp_geometry*.

	• *range* – A 2D area. • *rectangle* – A 2D area, aligned to the global axes. Besides *W* and *D*, also the parameters of the pre-configuration dialog (if existing) are available in the context of the shape creation. Workflow *Auto Fill* supports the shape types *range* and *geometry*. Workflows *Repeated Insertion, Selection, Position* and *Resize* support the shape type *geometry*. **Example**: *geometry*
Parameters (args)	The interactive behavior can be controlled further by a set of constants. The following constants are available: • *MinX* – Minimal width. • *MaxX* – Maximal width. • *DefX* – Default width. • *RasX* – Width raster. • *OffX* – Width offset for rasterization. • *PosX* – Explicit width. • *MinZ* – Minimal depth. • *MaxZ* – Maximal depth. • *DefZ* – Default depth. • *RasZ* – Depth raster. • *OffZ* – Depth offset for rasterization. • *PosZ* – Explicit depth. • *PosXZ* – Explicit width and depth. These constants are only available in the workflow Resize: • *MinY* – Minimal height. • *MaxY* – Maximal height. • *DefY* – Default height. • *RasY* – Height raster. • *OffY*– Height offset for rasterization. • *PosY* – Explicit height. In this context, width is equal to local x, the same applies to depth and local z as well as height and local y. Rules: • Either *RasN* or *PosN* or *PosXZ* can be defined. • *MinN*(*MaxN)* has higher priority than *RasN*. • If the closest rasterized value is smaller(bigger) than *MinN*(*MaxN*) it will be corrected to *MinN*(*MaxN*). • *DefN* will not be checked against *MinN, MaxN, RasN*. • *DefN* must be set greater than 0.0.

	• If *OffN* is not defined, it is considered to be 0.0. • As far as *MinN* and *MaxN* are defined, *MinN* < *MaxN* must hold. • If set, *MinN(MaxN)* must be greater than or equal to 0.0. • If *OffN* is not defined, it is considered to be 0.0. • If set, all values in *PosN* must be greater than 0.0. These parameters are only used in the workflows *Auto Fill* and *Resize*. **Example**: *DefX=0.8,MinX=0.8,MaxX=3.2,RasX=0.4*
Creation Mode (creation)	see Global Enumerator **Creation Mode** The creation mode is not needed in the workflows *Auto Fill* and *Resize*. **Example**: *article*
Article ID (article)	This entry references to an article defined in table *mp_articles* or a group defined in table *mp_groups*. The article ID is not needed in the workflows *Auto Fill* and *Resize*. **Example**: *desk800*
Variant Code (variant)	This entry provides additional variant information to be applied to the article after its creation. As this is optional, the entry can remain empty. If the article is a metatype, the meta-properties of this article can also be modified this way. The variant code is not needed in the workflows *Auto Fill* and *Resize*. **Example**: *[@man, @addon, [@GWidth, 1600]]*

mp_geometry

The table *mp_geometry* links to 3D and 2D geometries defined in external ODB data bases.

The layer MP_ANNOTATION in the 2D ODB can be used to mark annotation objects to be displayed if the associated geometry is high-lighted during shape selection.

Note, the MP selection spots are located at the layer MP_SPOT.

Shape ID (id)	This key defines a geometry entry in the table. Multiple lines with the same ID are illegal. The ID refers to the ID defined in the table *mp_shapes*. **Example**: *sh1*
ODB Type (odb_type)	This entry references the external ODB type by a fully qualified identifier. **Example**: *::metafactory::mt::deskW*

ODB Arguments (odb_args)	This entry provides optional parameters in order to implement parametric ODB objects. The arguments depend on the specific ODB data models. In case of non-parametric ODB models this entry can be left empty. **Example**: *W=1.2,IW=0.04*

mp_fillrules

The table *mp_fillrules* defines fill rules to be used by the workflow *Auto Fill.*

Fill Rule ID (id)	This key defines a fill rule. Typically multiple lines with the same ID do exist. **Example**: *fr1*
Condition (condition)	If specified, the condition must be true to activate the fill rule. Inside the condition the parameters *W* (width) and *D* (depth) can be used that represent the rectangular area of the workflow's shape. Empty conditions are interpreted as 'True'. **Example**: (W > 1.8) && (W < 2.2)
Creation Mode (creation)	see Global Enumerator **Creation Mode** **Example**: *article*
Article ID (article)	This entry references to an article defined in table *mp_articles*. **Example**: *desk800*
Variant ID (variant)	This entry provides additional variant information to be applied to the article after its creation. As this is optional, the entry can remain empty. If the article is a Metatype, the meta-properties of this article can also be modified this way. **Example**: *[@man, @addon, [@GWidth, 1600]]*
X Position (pos_x); Y Position (pos_y); Z Position (pos_z)	This value provides the offset in x;y;z direction of the article's origin regarding to the origin of the workflow, in Meter. If not set, the value is considered to be 0.0. **Example**: *1.0; 1.0; 1.0*
Rotation Direction (rot_dir)	If the article should be rotated relatively to the orientation of the workflow, this entry should provide the rotation axis (*x*, *y* or *z*). Otherwise the field can be left blank. **Example**: *y*
Rotation Value (rot_value)	If the article should be rotated relatively to the orientation of the workflow, this entry should provide the rotation offset around the axis specified above, in Degree. Otherwise the field can be left blank. **Example**: *90.0*

mp_articles

The table *mp_articles* defines article creation parameters to be used by shapes, fill rules or inside user exits.

Article ID (id)	This ID sets up an article creation entry. Multiple lines with the same ID are illegal. **Example**: *desk800*
Manufacturer ID (manufacturer)	This entry references the manufacturer name of the article. **Example**: *mf* (*MetaFactory*)
Program ID (program)	This entry references the program name the article belongs to. **Example**: *alu2* (*MetaFactory* program *Alu-2*)
Article Type (article_type)	This field is currently ignored. **Example**: -
Article Number (article_no)	This entry contains the article number (basic or final) for the article creation. **Example**: *MFA2800C1NF*
Variant Code (variant)	This optional entry contains an additional variant code (complete or partial) to be applied after the article creation. The syntax depends on the specific native article. If the article is a Metatype, the meta-properties of this article can also be modified this way. **Example**: *[@man, @addon, [@GWidth, 1600]]*

mp_groups

The table *mp_groups* defines a group of articles to be created by shapes or fill rules.

Group ID (id)	This ID sets up an article group entry. Typically multiple lines are provided for each key – one of them for each article. **Example**: *group_desk*
Creation Mode (creation)	see Global Enumerator **Creation Mode**. The creation mode *group* is not allowed here. **Example**: *article*
Article ID (article)	This entry references to an article defined in table *mp_articles*. **Example**: *desk800*
Variant ID (variant)	This entry provides additional variant information to be applied to the article after its creation. As this is optional, the entry can remain empty. If the article is a Metatype, the meta-properties of this article can also be modified this way. **Example**: *[@man, @addon, [@GWidth, 1600]]*

Position (pos_x); Y Position (pos_y); Z Position (pos_z)	This value provides the offset in x;y;z direction of the article's origin regarding to the origin of the workflow, in Meter. If not set, the value is considered to be 0.0. **Example**: *1.0; 1.0; 1.0*
Rotation Direction (rot_dir)	If the article should be rotated relatively to the orientation of the workflow, this entry should provide the rotation axis (*x*, *y* or *z*). Otherwise the field can be left blank. **Example**: *y*
Rotation Value (rot_value)	If the article should be rotated relatively to the orientation of the workflow, this entry should provide the rotation offset around the axis specified above, in Degree. Otherwise the field can be left blank. **Example**: *90.0*

mp_attpt

The table *mp_attpt* defines attachment points to be used for geometric arrangements and identification of interactive manipulation targets. Attachment points are defined for shapes. Therefore, the shape ID is the primary key to the table of attachment points.

Shape ID (id)	This key references a shape in table *mp_shapes*. Typically more than one row exist for one and the same ID. **Example**: *sh1*
Key (key)	The entry specifies a symbolic value for identification purposes. Usually the same keys are used as in the context of the Metatypes. **Example**: *MFRF*
Direction (direction)	The entry specifies a symbolic value for identification purposes. Usually the same keys are used as in the context of the Metatypes. **Example**: *R*
Condition (condition)	The condition entry provides information about the attachment point in the following sense: • *RS_R* – The attachment point should be used for right-sided resize operations. • *_RS_R* – The attachment point should be used as opposite point for right-sided resize operations. • *SPOT_R* – The attachment point should be used as an explicit position for the interactive manipulation target to select a shape. If not set, *RS_R* should be used. *SPOT_R* resp. *RS_R* must be visualized by the run-time environment. **Example**: *RS_R*
X Position (pos_x) ; Y Position	These columns contain the local coordinates of the position of the attachment point, in Meters.

(pos_y) ; Z Position (pos_z)	**Example**: *0; 0.72; 0*
Y Rotation (rot_y)	This entry defines the rotation around the y axis in Degree, relative to the parent object. **Example**: *90.0*

mp_attptpairs

The table *mp_attptpairs* defines matching attachment points. Attachment point pairs are used for instance in the *Repeated Insertion* workflow to describe matching pairs of attachment points. Such pairs are defined specifically for shape sets. Therefore, the shape set ID is the primary key to the table of attachment point pairs. For one shape set, only one attachment point pair can be defined.

Shape Set ID (shape_or_set)	This key references a shape set from table *mp_shapesets*. Typically more than one row exists for one and the same ID. **Example**: *ss1*
Key (key)	The entry gives the 'left side' of the attachment point pair. This is the used key of the already existing object. **Example**: *MFLF*
Opposite Key (opposite)	The entry gives the 'right side' of the attachment point pair. This represents one (there can be more) key of the next object to be created. **Example**: *MFRF*

mp_info

The table *mp_info* implements meta information about the MP data set.

Key (key)	This entry names an information tag and must be one the ones defined below. **Example**: *major*
Value (value)	This entry provides the value associated to that key. **Example**: *1*

The following meta properties and values are defined. Optional keys are marked with (*).

- *major* - Defines the major number of the underlying MP version, currently: 1. If unset it's considered as 1.
- *minor* - Defines the minor number of the underlying MP version, currently: 17. If unset it's considered as 11.

mp_spots

The table *mp_spots* is used to modify the predefined graphical representation of selection spots (see *mp_attpt* condition SPOT_R). This representation is defined in external ODB data bases.

Shape ID (id)	This key references a shape in table *mp_shapes*. **Example**: *sh1*
ODB Type (odb_type)	This entry references the external ODB type by a fully qualified identifier. **Example**: *::metafactory::mt::spot_circle*
ODB Arguments (odb_args)	This entry provides optional parameters in order to implement parametric ODB objects. The arguments depend on the specific ODB data models. In case of non-parametric ODB models this entry can be left empty. **Example**: *Radius=0.1,Filled=@yes*

5 Implementation issues

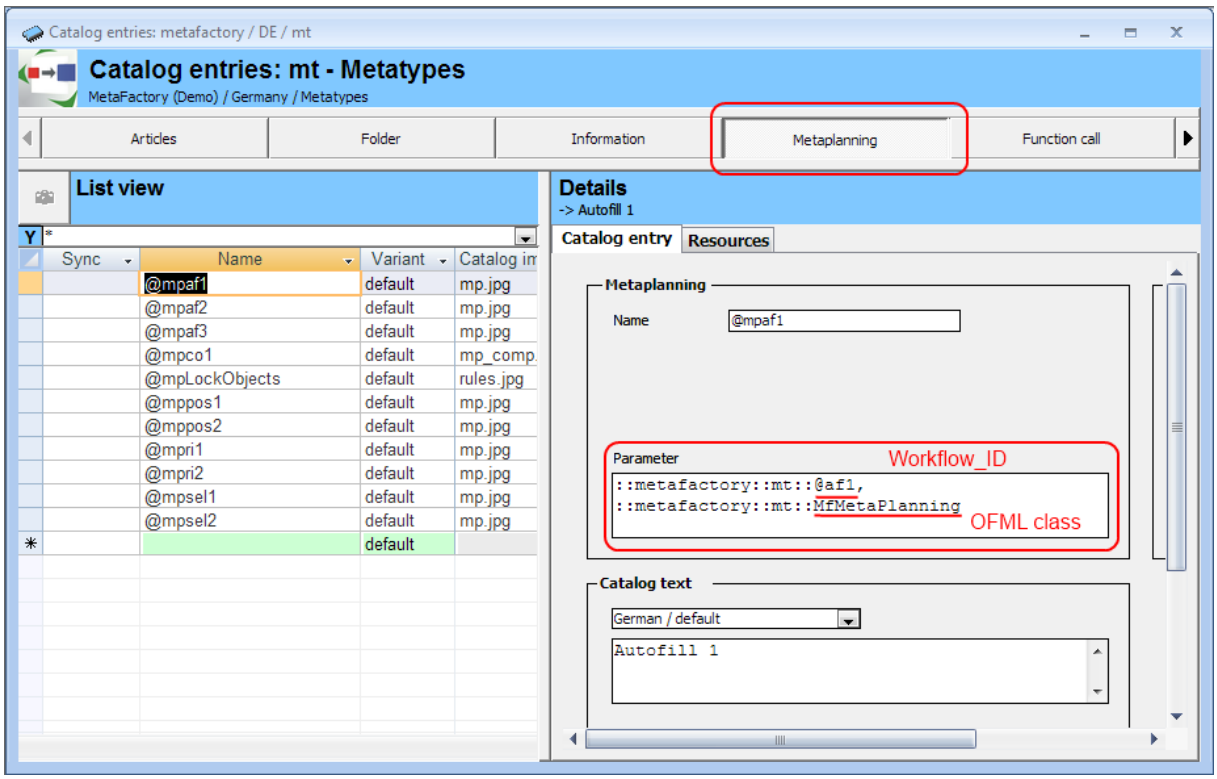
The Metaplanning database has to be located in the OFML data directory of a series. In some cases it is useful to use a separate product line for the Metaplanning data. The name of the EBase database has to be the name of the series with the suffix "_mp".
e.g. `tables_mp.ebase`

The text resources must be resolved in the resource files according to the OFML standard (*global_de.sr*, *global_en.sr*, etc.).

A Metaplanning workflow has to be linked in the OFML catalog. There is a special insertion type for Metaplanning workflows: *MP*
e.g. `@mpaf1;;;;MP;15;`

Furthermore, the Metaplanning resource key has to be specified. It consists of
<manufacturer_name>::<series_name>::@<Workflow_ID>,
<manufacturer_name>::<series_name>::<OFMLClass for User Exits>
e.g. `@mpaf1;;;MP;::metafactory::mt::@af1,::metafactory::mt::MfMetaPlanning;`

This linking can also be done with the OAS Module of pCon.creator.



6 History

This history describes all modifications relevant to this specification. All further changes are documented in the history provided with the Metatype implementation.

MP 1.17.0-0

- [NEW] New User Exit *getNextChildWf()*.
- [NEW] New User Exit *getObjDimensions()*.
- [NEW] New User Exit *setObjDimensions()*.
- [NEW] New User Exit *getShapeArgs()*.
- [NEW] New User Exit *terminateWorkflow()*.
- [NEW] New workflow type *Resize*.

MP 1.16.0-0

- [NEW] New workflow type Position.

MP 1.15.0-0

- [NEW] New table *mp_spots*.
- [NEW] New table *mp_groups*.

MP 1.13.0-2

- [NEW] Revised description of *mp_workflows::next_pc*.

MP 1.13.0-1

- [NEW] Special layer *MP_SPOT* for the MP selection spots

MP 1.13.0(-0)

- [NEW] Special layer *MP_ANNOTATION* to be used for 2D ODB geometries
- [NEW] New column *position* in table *mp_composites*

- [NEW] New workflow type *Selection*.
- [NEW] New termination policy *goto_<No>* for composite workflows.

- [NEW] New User Exit *customizePreConfiguration()*
- [NEW] Parameters of pre-configuration dialog available in shape creation context.
- [NEW] New creation mode *item* to create basket items.
- [NEW] New creation mode *metatype* as addition to *article*.
- [NEW] New table *mp_info* with initial keys *major* and *minor*.
- [NEW] New table *mp_numprops*.
- [NEW] New column Property Mode in table *mp_preconfig*
- [NEW] New condition *SPOT_R* for table *mp_attpt*

- [NEW] User Exit *getNextPosRI()*

- [MOD] *mp_workflows*: workflow IDs may not start with an underscore
- [MOD] *mp_geometry*, ODB Args are also considered during workflow *Repeated Insertion*
- [NEW] Workflow *Composite* plus table *mp_composites*
- [NEW] User Exit *finishWorkflow()*

MP 1.11.0(-0)