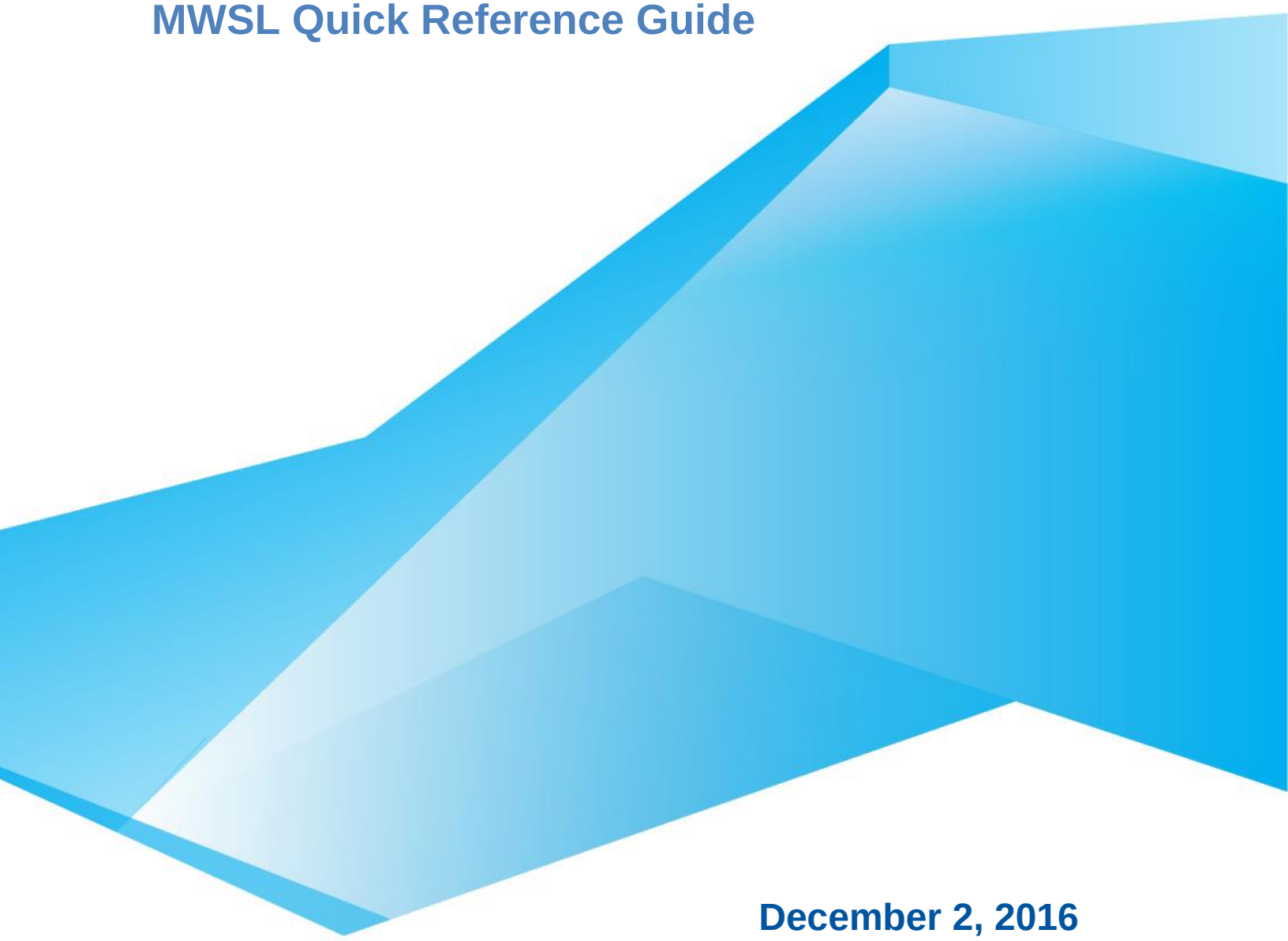
Deltek Know more. Do more.™

Deltek Maconomy 2.3 GA

MWSL Quick Reference Guide



December 2, 2016

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published December 2016.

Contents

Overview

This document is a quick reference to MWSL. It briefly describes all of the tags and associated attributes.

Attributes of a given tag are described using a table like the following.

Attribute Name	Type	Usage
Attr1	*Type of attr1*	This attribute is used to…
Attr2	*Type of attr2*	Indicates…

The referred types of the attributes can be one of the following.

Attribute	Description
Boolean	A Boolean attribute is true or false.
Expression	An expression in the general Expression Language (EL).
Key	A string that is case-insensitive and never exposed to an end user. Use it for references (internally or from other parts in the spec/other specs). A key must be in the form: (_a-zA-Z)(_a-zA-Z0-9)*, for example: “myName1.”
NsKey	A name-spaced string that is case -insensitive and never exposed to any end user. Use it for reference (internally or from other parts in the spec/other specs). An NsKey must be in the form: (Key:)?(Key), for example: “myNS:myName2” or “myName3.”
Display	A string that is meant to be displayed to an end user (and that is therefore localized). It can never be used for reference.
Id	A case-sensitive string that is used to reference items in environments that are not controlled by Maconomy. It is never exposed to an end user.
Enumeration	An enumerated set of string values.

Preamble Tags

<Definitions>-tag

Use the <Definitions>-tag to contain various forms of definitions that are available when parsing the workspace. Only functions can be specified.

There are no attributes for the <Definitions>-tag.

<Function>-tag

The <Function>-tag is embedded within the <Definitions>-tag. This tag declares a named function that can be accessed in expressions throughout the workspace.

Attribute Name	Type	Usage
name	Key	The name of the function. Use this name to refer to the function from within expressions.
type	Enumeration	This is the *return type* of the function. The default type of expression is Boolean, but functions of other types can be created.
value	Expression	This is the *function body*. The function body is itself an expression and may refer other functions that are declared above or functions that are available as “built-in” functions or functions that are “plugged in.”
Parameters	Key list	Here you specify the formal names of the parameters, (the *input parameters)* for the function. When invoking a function, you must invoke it with as many parameters as are declared in this list.

Example

```
<Function name=”mult”
          type=”real”
          value=”x * y”
          parameters=”x y”/>
```

This defines a “mult” function that returns a value of the type real. It takes two parameters, “x” and “y,” and returns these two values multiplied.

Examples of invoking this function are:

```
2 + mult(2, 17.3)

mult(mult(6,4), mult(2,6))
```

Component Tags

<Filter>-tag

The <Filter>-tag specifies the presence of a filter pane at this level in the workspace. The filter pane can contain bindings to other panes, and these can be organized in different compositional structures (Expansions, Assistants). “Initial filters” are compactable. An initial filter is a filter that either has no parents, or only has initial filters as parents. Also, an initial filter must have either no siblings or filter-pane siblings only. In this case, the filter is treated as part of “the initial data finding.” Such filters (tab rows) can be “compacted,” thereby only showing the selected filter in a very compacted mode. Compacting is not supported by the client.

A filter that is not bound by any other pane can show any record that is referred by the where clause of the default cursor and corresponding dialog.

Attribute Name	Type	Usage
source *Not available if filter is bound using a <With>-binding*	NsKey	Indicates the name of the dialog that is used to define the content of the filter. For example, if the dialog is “InvoiceSelection,” the filter shows jobs (this is the entity that is shown in the upper part of that dialog). It also unconditionally applies the filtering that is defined by the where-clause of the upper pane cursor of that dialog. (In this case, meaning that it is not possible to see jobs that are not able to be invoiced.)
Title	Display	The title of the tab that represents access to the filter pane. If not specified, the title of the associated MDML layout is used, or if that title is undefined the default title that is specified for that FilterPane in DDL is applied.
Layout	NsKey	Name of the MDML file that contains the layout that should be applied for this pane (excluding “.mdml.xml”). If this value is an empty string, no layout is applied, and a blank pane is displayed. (This is useful to quickly make a mockup workspace.) If this attribute is not specified, the layout called *dialogName*.mdml.xml is used. If the pane is bound by a <With>-binding, the layout value from its parent is used as the default when no value is explicitly specified.
view	Key/Expression	Name of the layout view to apply. In a layout file, each type of pane can have many different layout *views* defined. Laying out a pane requires *one* view. The first view that has the name specified by this attribute is applied. If no value is specified, the first view (for the relevant pane) is applied. It is possible to give *arguments* to a view (provided that the view is declared with corresponding parameters in MDML). In this case, you can add a

Attribute Name	Type	Usage
		number of expressions in a comma-separated list, for example: `view="myView(true, 33 - 2, currentDate())"` If the pane is bound by a <With>-binding, the view value from its parent is the default if no value is explicitly specified.
Name	NsKey	The internal name of the pane in *this* workspace. Currently unused, but may eventually be used to address a specific pane in a workspace from, for example, links, notifications, and so on.
pluginId	Id	The ID of the plug-in that implements the filter pane to apply. If left unspecified, the default filter-implementation is used.

<Card>-tag

The <Card>-tag specifies the presence of a card pane at this level in the workspace. The card pane can contain bindings to other panes, and these can be organized in different compositional structures (for example, Expansions and Assistants).

A card that is not bound by any other pane shows a *random record* from the set of records identified by the where clause of the default cursor of the corresponding dialog. This is useful for dialogs that are capable of showing only one record, such as SystemInformation. The Card tag is used to reference normal card-parts as well as "action cards."

Attribute name	Type	Usage
source *Not available if card is bound using a <With>-binding*	NsKey	Indicates the name of the dialog that is used to define the content of the card. For example, if the dialog is "TimeSheets," the card may show time sheet headers plus variables that are defined for the card part of the dialog "TimeSheets."
Title	Display	The title of the tab that represents access to the card pane. If not specified, the title of the associated MDML layout is used, or if that title is undefined, the default title specified for that CardPane in DDL is applied.
layout	NsKey	Name of the MDML file that contains the layout that should be applied for this pane (excluding ".mdml.xml"). If this value is the empty string, no layout is applied and a blank pane is displayed. (This is useful to quickly make a mockup workspace.) If this attribute is not specified, the layout called *dialogName*.mdml.xml is used. If the pane is bound by a <With>-binding, the layout

Attribute name	Type	Usage
		value from its parent is the default if no value is explicitly specified.
View	Key/Expression	Name of the layout view to apply. In a layout file, each type of pane can have many different layout *views* defined. Laying out a pane requires *one* view. The first view that has the name specified by this attribute is applied. If no value is specified, the first view (for the relevant pane) is applied. It is possible to give *arguments* to a view (provided that the view is declared with corresponding parameters in MDML). In this case, you can add a number of expression in a comma-separated list, for example: `view="myView(true, 33 - 2, currentDate())"` If the pane is bound by a <With>-binding, the view value from its parent is the default if no value is explicitly specified.
Name	NsKey	The internal name of the pane in *this* workspace. It is currently unused, but may eventually be used to address a specific pane in a workspace from, for example, links, notifications, and so on.
pluginId	Id	The ID of the plug-in that implements the card pane to apply in this instance. If left unspecified, the default filter-implementation is used.
hidden	Expression	If the expression evaluates to true, this card (and all of its children) are not visible in the workspace; they are hidden. This means they may as well not be there. The expression is evaluated in the context of the current card. This means that all fields, functions, and expressions that are exposed to the card can be used for calculation on the visibility of the card. For this particular attribute, three special functions are available: **hasNoSeed()** The `hasNoSeed()` function is true if the pane does not have any data, *not even "0 rows."* Basically, this means that the underlying dialog pane cannot be read using the foreign key, for example if the foreign key is not enabled, if the dialog cannot show the designated record, or if the record does not exist in the database.

Attribute name	Type	Usage
		hasNoRecords() The `hasNoRecords()` function is true if the pane shows 0 rows, *including the case* where the dialog pane (a table) *is* associated with an existing record (from the card pane), but where there are just no rows in the table. Thus, the `hasNoRecords()` function should be used with great caution, because it might prevent a user from populating the table (because the "insert" or "add" row are not accessible when the pane is not shown). Thus, the `hasNoRecords()` function should only be used in cases where it is never possible to add new rows to a table. `hasNoRecords()` covers *all* cases that are covered by `hasNoSeed()` *plus* the case where data has been read; there just happen to be 0 rows in the result. **hasNoChildren()** The `hasNoChildren()` function is true if the pane does not have any children. The number of children is dynamically calculated from the hidden-attribute of all of the children of the current pane. This means that if the current pane has a number of children, which are *all* dynamically hidden from their state, the parent (current) pane is also hidden.

<Table>-tag

The <Table>-tag specifies the presence of a table pane at this level in the workspace. A table pane always corresponds to the table part of a card-table dialog. The table pane can contain bindings to other panes, and these can be organized in different compositional structures (for example, Expansions, Assistants). Technically, a table pane is always bound by referring the record in the *upper pane* of the same dialog.

A table that is not bound by any other pane shows the table part of a *random upper-pane record* that is identified by the where clause that defines the upper-pane cursor in the corresponding dialog. This can be useful for card/table-dialogs that are only capable of showing one upper-pane record.

Attribute Name	Type	Usage
source *Not available if table is bound using a <With>-binding*	NsKey	Indicates the name of the dialog that is used to define the content of the table. For example, if the dialog is "TimeSheets," the table shows time sheet lines plus variables defined for the table part of the dialog "TimeSheets."
title	Display	Represents access to the table pane. If not

Attribute Name	Type	Usage
		specified, the title of the associated MDML layout is used, or, if that title is undefined, the default title specified for that TablePane in DDL is applied.
layout	NsKey	Name of the MDML file that contains the layout that should be applied for this pane (excluding ".mdml.xml"). If this value is the empty string, no layout is applied. A blank pane is displayed. (This is useful to quickly make a mockup workspace.) If this attribute is not specified, the layout called *dialogName*.mdml.xml is used. If the pane is bound by a <With>-binding, the layout value from its parent is the default if no value is explicitly specified.
view	Key/Expression	Name of the layout view to apply. In a layout file, each type of pane can have many different layout *views* defined. Laying out a pane requires *one* view. The first view that has the name specified by this attribute is applied. If no value is specified, the first view (for the relevant pane) is applied. It is possible to give *arguments* to a view (provided that the view is declared with corresponding parameters in MDML). In this case, you can add a number of expressions in a comma-separated list, for example: `view="myView(true, 33 - 2, currentDate())"` If the pane is bound by a <With>-binding, the view value from its parent is the default if no value is explicitly specified.
name	NsKey	The internal name of the pane in *this* workspace. This is currently unused, but may eventually be used to address a specific pane in a workspace from, for example, links, notifications, and so on.
pluginId	Id	The ID of the plug-in that implements the table pane to apply in this instance. If left unspecified, the default filter-implementation is used.
hidden	Expression	If the expression evaluates to true, this table (and all of its children) are not visible in the workspace; they are hidden. This means that they may as well not be there. The expression is evaluated in the context of the current table. This means that all fields, functions, and expressions that are exposed to the table can be used for calculation of the visibility of the table.

Attribute Name	Type	Usage
		For this particular attribute, three special functions are available. **hasNoSeed()** The `hasNoSeed()` function is true if the pane does not have any data, *not even "0 rows."* Basically, this means that the underlying dialog pane cannot be read using the foreign key, for example if the foreign key is not enabled, if the dialog cannot show the designated record, or if the record does not exist in the database. **hasNoRecords()** The `hasNoRecords()` function is true if the pane shows 0 rows, *including the case* where the dialog pane (a table) *is* associated with an existing record (from the card pane), but where there are just no rows in the table. Thus, the `hasNoRecords()` function should be used with great caution, because it might prevent a user from populating the table (because the "insert" or "add" row is not accessible when the pane is not shown).The `hasNoRecords()` function should only be used in cases where it is not possible to add new rows to a table. `hasNoRecords()` thus covers *all* cases that are covered by `hasNoSeed()` *plus* the case where data has been read; there just happen to be 0 rows in the result. **hasNoChildren()** The `hasNoChildren()` function is true if the pane does not have any children. The number of children is dynamically calculated from the hidden-attribute of all of the children of the current pane. This means that if the current pane has a number of children, *all* dynamically hidden from there state, the parent (current) pane is also hidden.

<Hidden>-tag

A hidden tag corresponds to a card pane that is *not* shown to a user. This can be handy to "secretly" bind two dialog-panes that are not directly related. For example, if entity E1 points to a record in entity E2, and E2 points to a record in entity E3. Then suppose that there are three dialogs: D1 (containing E1), D2 (containing E2), and D3 (containing E3). You want a workspace that contains a pane that shows D1 followed directly by one that shows D3 (where the record that is displayed is the one that is identified by the record in D2, which was referred by the record in D1). In this case, a workspace like the following can be used:

```
<Card dialog="D1">
  <Bind foreignKey="E1toE2">
    <Hidden dialog="D2">
      <Bind foreignKey="E2toE3">
        <Card dialog="D3"/>
      </Bind>
    </Hidden>
  </Bind>
</Card>
```

This yields a workspace that shows a card pane at the top (D1) followed by another card pane (D3).

A hidden pane that is not bound by any other pane behaves just like a corresponding <Card> would behave, except that it is not shown.

Attribute Name	Type	Usage
Source	NsKey	Indicates the name of the dialog that is used to define the content of the hidden card. For example, if the dialog is "TimeSheets," nothing is shown, but you may create bindings from this pane using all foreign keys that are defined for the card part of the Time Sheets dialog.
Name	NsKey	The internal name of the pane in *this* workspace. Currently unused, eventually it may be used to address a specific pane in a workspace from, for example, links, notifications, and so on.

<Workspace>-tag (embedded)

The <Workspace>-tag, in embedded form, is used to "include" the contents of another workspace specification into this one. It can be used in situations where a pane can be referred. At that position, the content of the referred workspace specification (excluding its outer-most workspace tag) is inserted into this workspace. The embedded workspace can recursively embed other workspace specs.

Attribute Name	Type	Usage
source	NsKey	The name of the workspace spec to "include" at this place.
Name	NsKey	Currently unused, but may be used in the future to reference this workspace inclusion.

<Section>-tag

The <Section>-tag marks a "titled" button/tab that has no direct content (for example, its content is defined by its inner tags; alone it has no content). See the <Formation>-tag for more details.

Because a segment does not hold any data by itself, it is bypassed in the data binding. Therefore, a segment must contain either a formation or a binding; the binding relates to the nearest pane that precedes the segment. For example, consider the following specification:

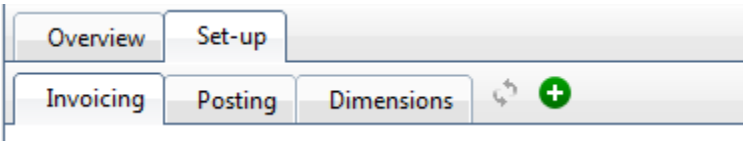
```
<Filter dialog=”Jobs”>
  <Formation>
    <Segment title=”Home”>
      <Bind> … </Bind>
      <Bind foreignKey=”MainJob”> … </Bind>
    </Segment>

    <Segment title=”Budget”>
      <Bind> … </Bind>
    </Segment>
  </Formation>
</Filter>
```

Here the implicit “primary” bindings in segments “Home” and “Budget” refer to the current record in the Jobs-filter. Similarly, the “MainJob” binding refers to the current record in the Jobs filter.

It is also possible to have “sections” (empty tabs) beside normal panes (for example, outside of <Formation>s). In this case, the section is represented as an empty tab with its expansion following directly.

In the following, you can see a section “Set-up” where tabs follow directly below.



Attribute Name	Type	Usage
name	Key	The internal name of the section in *this* workspace. This is currently unused, but may eventually be used to address a specific segment in a workspace from, for example, links, notifications, and so on.
title	Display	The title of the section displayed in the button/tab.
hidden	Expression	If the expression evaluates to true, this section (and all of its children) are not visible in the workspace; they are hidden. This means that they may as well not be there. The expression is evaluated in the context of the current section. This means that all fields, functions, and expressions that are exposed to the section can be used for calculation on the visibility of the section. For this particular attribute, three special functions are available: **hasNoSeed()** The `hasNoSeed()` function is true if the pane does not have any data, *not even “0 rows”*. Basically, this means that the underlying dialog pane cannot be read using the foreign key, for example if the foreign key is not enabled, if the dialog cannot show the designated record, or if the record does not exist in

Attribute Name	Type	Usage
		the database. **hasNoRecords()** The `hasNoRecords()` function is true if the pane shows 0 rows, *including the case* where the dialog pane (a table) *is* associated with an existing record (from the card pane), but where there are just no rows in the table. Thus, the `hasNoRecords()` function should be used with great caution, because it might prevent a user from populating the table (because the “insert” or “add” row is not accessible when the pane is not shown). Thus, the `hasNoRecords()` function should only be used in cases where it is never possible to add new rows to a table. `hasNoRecords()` thus covers *all* cases that are covered by `hasNoSeed()` *plus* the case where data has been read; there just happen to be 0 rows in the result. **hasNoChildren()** The `hasNoChildren()` function is true if the pane do not have any children. The number of children is dynamically calculated from the hidden-attribute of all of the children of the current pane. This means that if the current pane has a number of children, which are *all* dynamically hidden from there state, the parent (current) pane is also hidden.

Container Tags

<Workspace>-tag (outermost level)

The <Workspace>-tag marks the contents of the workspace.

Attribute Name	Type	Usage
name	NsKey	Indicates the name of the workspace. It must be identical to the file name (leaving out ".mwsl.xml"). A namespace defines a sub-folder in the workspace-folder. (For example, MyNamespace:MyFile must be stored in: mynamespace/myfile.mwsl.xml.)
title	Display	The title of the workspace. This tag is largely unused. (The title of the workspace is taken from the menu spec.) If a workspace is loaded by other means than the menu spec., this title is used. In time, the menu spec. could be compiled from the titles that are defined by the workspace specs.

<Expansions>-tag

The <Expansions>-tag groups together a number of panes. They are shown as a number of tabs. The content of the expansion direction is assumed to indicate the primary purpose of a workspace, which is reflected by current and future defaults.

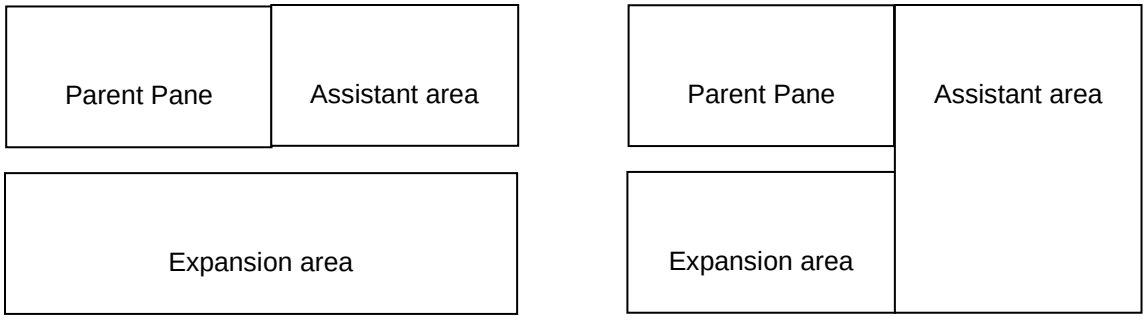
Attribute Name	Type	Usage
name	Key	Indicates the name of the expansion collection. This is currently unused, but may be used in the future to reference a specific expansions collection.
horizontal	Boolean	The default is false. (This is currently true for expansions following an initial filter. This will be changed in the future.) If true, the expansion tabs are displayed to the *right* of the parent pane. Otherwise, the expansion tabs are displayed *below* the parent panes.
minimized	Boolean	The default is false (except for the first expansion that does not contain an initial filter). If true, the expansions are minimized by default (so not directly visible). Instead a reveal button is displayed.
parentSize	Enumeration	This attribute specifies the ratio between this expansion and its parent pane by defining the relative size of the *parent pane*. Possible values are:

Attribute Name	Type	Usage
		`tiny, small, medium, large, huge, vast`
hidden	Expression	Determines the default expression for the hidden attribute for panes in this expansion.

<Assistants>-tag

Assistants are used to group a number of panes. They are shown as a number of tabs. The content of the assistant direction is assumed to contain information that is “occasionally needed” (in contrast to Expansions, which are considered “prime content”). This fact is reflected by current and future defaults. Apart from the defaults, there are no differences between what can be done with assistants and what can be done with expansions.

Attribute Name	Type	Usage
name	Key	Indicates the name of the assistant collection. This is currently unused, but may be used in the future to reference a specific assistants collection.
horizontal	Boolean	The default is true. (Currently, false is for assistants that follow an initial filter; this will be changed in the future.) If true, the assistant tabs are displayed to the *right* of the parent pane. Otherwise, the expansion tabs are displayed *below* the parent panes.
minimized	Boolean	The default is true. If true, the assistants are minimized by default (that is, not directly visible). Instead a reveal button is displayed.
enlarged	Boolean	The default is false. This attribute has effect *only* when the parent pane has *both* assistants and expansions. If true, the assistants are space-prioritized over the expansions. If false, the expansions are space-prioritized.
parentSize	Enumeration	This attribute specifies the ratio between this assistant and its parent pane by defining the relative size of the *parent pane*. Possible values are: `tiny, small, medium, large, huge, vast`
hidden	Expression	Determines the default expression for the hidden attribute for panes in this assistant.

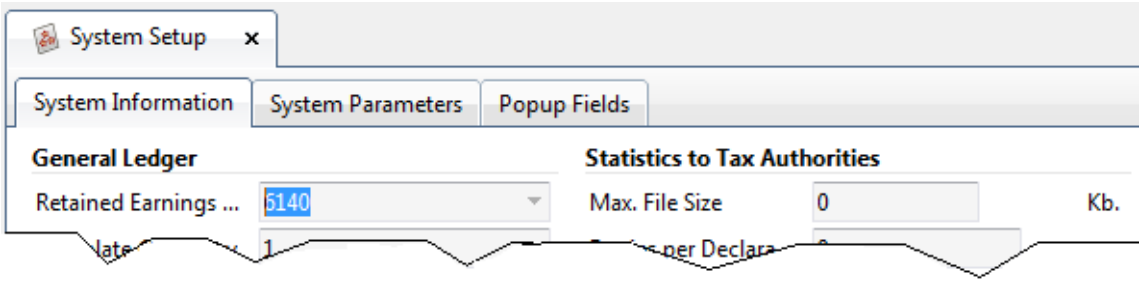


Expansion is space prioritized

Assistant is space prioritized

Initial Collection of Tabs

The “root” of a workspace can be thought of as “an initial expansion.” Thus, it is possible to initially specify a number of parallel structures. Notice the difference between initial parallel structures and a formation: A formation is a collection of “title buttons” (that is, no pane content) rendered as “boxes,” whereas an initial parallel structure is a collection of panes (each of which has a tab that represents access to that pane).



<Formation>-tag

A formation represents a collection of “title buttons” or “empty tabs.” Visually, formations are rendered as a “ribbon that contains rectangular titled buttons” (specified by <Section>-tags). There is *no* content that is directly associated with any of these title buttons. The only thing that can follow is an Expansions-collection (for example, a collection of panes, each represented with a tab), another formation, or a *Mount*-connection. (See <Mount>-tag.) Formations can be used to offer a selection of what appears to be a “sub workspace” or can be seen as a “sub menu.”

Attribute Name	Type	Usage
name	Key	Indicates the name of the formation collection. This is currently unused, but may be used in the future to reference a specific formation collection.

Connection Tags

Connection tags are used to define how two consecutive panes are bound together by their data, in other words, how the parent pane defines the content of the following pane.

<Mount>-tag

The mount connection defines that the parent binding has *no influence* on the contents of the following pane. This implies that anything following a <Mount>-tag is populated in the same way as it would be if it were the first pane in a workspace. A workspace always implicitly starts with a <Mount>-binding. <Mount> is often used to specify the “beginning” in segments inside an initial formation, or to connect to panes that do not have a formalized key (such as action cards). Also, it is possible to start “anew” inside a workspace by using the <Mount>-connection.

There are currently no attributes for the <Mount>-tag.

<Bind>-tag

The bind connection is used to bind a parent pane to a *single record*. The content of the following pane is defined by *one single* record.

Attribute Name	Type	Usage
name	Key	Indicates the name of the connection. It is currently unused, but may be used in the future to reference fields from child panes. If unspecified, the name defaults to the name of the foreign key specified by the foreignKey attribute.
foreignKey	Key	The name of the foreign key to use for this binding. The referred name must exist in the parent pane, and it must point to the entity that is contained by the child pane. If you just want the “record itself,” use the foreign key that is by convention named “primary.”

<Restrict>-tag

The restrict connection is used to connect filter panes. The restriction denotes a set of records that can be shown in the child filter pane based on the record that has focus in the parent filter pane.

For the casual eye, there are two different uses of the restrict connection: one where a foreign key from *both* the parent and the child pane is specified, and one where only a foreign key from the child filter pane is specified.

Example: Foreign Key Specified on Child Pane *only*

The parent pane shows customers, and the child pane shows a job. The foreign key from the *child pane* (referred to as the “reversed foreign key” because it points in the “reverse” direction of the data flow) is “customer referred by a job.” This means that any record that is shown in the *child* pane must point out (using that foreign key) exactly the record that has focus in the parent

pane. In this case, it means that all jobs that are shown in the child pane must point to the customer that is selected in the parent pane.

Example: Foreign Key Specified on *both* Child Pane and Parent Pane

The parent pane shows “customer,” and the child pane shows “employees.” The foreign key from the *child pane* (in other words, the “reversed foreign key”) points out a zip code. The foreign key from the *parent pane* (in other words, the “foreign key”) also points out a zip code. Both foreign keys must point to the same entity. This means that any record that is shown in the *child* pane must point out the *same* zip code that is pointed out by the selected record in the *parent pane*. The “zip code” is used as a “stepping stone” between the parent and the child pane entities. In this case, it means that selecting a customer, the second pane shows employees that live close to that customer (in other words, in the same zip code area).

The two uses are actually the same: in the first case (in other words, only specifying a “reversed foreign key”), the parent pane always implicitly points to itself.

Attribute Name	Type	Usage
name	Key	Indicates the name of the connection. This is currently unused, but may be used in the future to reference fields from child panes. If unspecified, the name defaults to the name of the foreign key specified by the reversedForeignKey attribute.
reversedForeignKey	Key	The name of the foreign key that points from the child pane to the record/entity pointed out by the foreign key from the parent pane. If the foreignKey attribute is left unspecified, it defaults to “the current record in the parent pane,” meaning that the reversedForeignKey should point out the current record in the parent pane in this case.
foreignKey	Key	The default is primary (in other words, the same record as is found in the parent pane). The name of the foreign key to use for this binding. The referred name must exist in the parent pane, and it must point to the same entity that is pointed to by the reversed foreign key.

<With>-binding

<With>-bindings are used to connect panes from the same source container (for example, the same dialog). Basically it means that whatever connection is used internally in the dialog to populate data in several panes is used. It also means that panes that are bound together using the <With>-binding share variable state space. This is most easily understandable for card/table dialogs: the content of the *table* part is determined intrinsically by the application source code for the dialog. In particular, the content of the table pane may depend on the value of *variables* that are held by the *card pane*. Thus, to obtain the table that intrinsically belongs with a particular card-record, you must use the <With>-binding. In many ways, a <With> binding is similar to <Bind foreignKey=”primary”>, but the very important difference is that using the <With> binding, you are guaranteed to get the pane that belongs with the one to which it is <With>-bound.

As an oddity, binding a card and a table pane from the same dialog together using <Bind foreignKey=”primary”> guarantees that if the table content depends on *variables* that are manipulated in the card, the table does *not* reflect these values.

You can *only* bind panes together using <With> if they originate from the same container source (dialog). In fact, the syntax makes certain of that. Thus, if you want to bind two panes of different dialogs together you *cannot* use <With>. You must use <Bind foreignKey=”…”> or one of the other bindings.

Typically, when you have panes from the same dialog, you want to bind them together using <With>.

For dialogs, it is supported to connect Card, Table, and Filter using <With>.

There are currently no attributes for the <With>-binding.

<Through>-binding

The <Through>-binding is used when you insert a <Section> (empty tab) in the middle of a chain of panes that data-wise depend on each other. Basically it means that the data context that is immediately *above* is passed “through” the section and, thus, is available for the children of the section.

An example is:

```
<MWSL xmlns="http://www.deltek.com/ns/mwsl" version="0.24">

  <Workspace name="MyWorkspace">

    <Filter source=”Jobs”>
      <With>
        <Card/>
      </With>
      <Through>
        <Section title=”This is an empty tab”>
          <With>
            <Card/>
          </With>
          <Bind foreignKey=”ProjectManagerNumber_Employee”>
            <Card source=”Employees”/>
          </Bind>
        </Section>
      </Through>
    </Filter>

  </Workspace>

</MWSL>
```

In this example, you start out with a filter of jobs. In the following, there is first the card of the dialog “Jobs” that shows the job that is selected in the filter. Also below the filter, there is an empty tab. And below that empty tab there *again* is access to the data that is selected in the filter. This enables the showing of that job card-pane again, as well as an employee card that shows the project manager of the selected job.

There are currently no attributes on the <Through>-binding.

Deltek is the leading global provider of enterprise software and information solutions for professional services firms, government contractors, and government agencies. For decades, we have delivered actionable insight that empowers our customers to unlock their business potential. Over 14,000 organizations and 1.8 million users in approximately 80 countries around the world rely on Deltek to research and identify opportunities, win new business, optimize resource, streamline operations, and deliver more profitable projects. Deltek – Know more. Do more.®

deltek.com