byron

Byron Informatik AG
Lohweg 6
CH-4054 Basel
Tel. +41 (0)61 690 96 00

byron@byron.ch
www.byron.ch

Byron/BIS – Technische Produktinformation
Toolkonfiguration

Building Information Systems
Software-Entwicklung und -Projekte

Inhaltsverzeichnis

1 Einführung

Die Konfiguration von Tools (Werkzeuge) wurde mit der Version 4.1 vereinheitlicht. Die Werkzeuge sind nicht mehr programmiert, sondern konfiguriert. Die Konfiguration erfolgt in XML und umfasst den Aufbau des Tools, Menu, Drag&Drop Operationen, alles was ein Tool ausmacht.

Dieses Dokument, bzw. die aktuelle Version davon befindet sich:
www.byron.ch/downloads/dokumente/ByronBIS_ToolKonfiguration.pdf

An vielen Stellen dieser Konfiguration wird Filternavigation verwendet. Das Referenzdokument befindet sich:
www.byron.ch/downloads/dokumente/FilterNavigationDescription.pdf

Eine vereinfachte Version auf deutsch:
www.byron.ch/downloads/dokumente/FilterNavigation_Einführung.pdf

1.1 Terminologie

1.1.1 Tool / Werkzeug

Ein Werkzeug ist ein Objekt in BIS, welches geöffnet und damit gearbeitet werden kann.

1.1.2 Application / Anwendung

Bis jetzt war ein Werkzeug eine Anwendung. Neu kann ein Werkzeug auch mehrere Anwendungen umfassen. Der Benutzer kann im Werkzeug die Anwendung umschalten.

Die Analogie: In Outlook (Tool) kann ich Mails und Termine (2 Anwendungen) bearbeiten.

1.1.3 Application Element / Anwendungselement

Die Anwendung wird aus Elementen zusammengesetzt. Ein Explorer typischerweise aus einem Baum links und einer Liste (Grid) rechts. Diese Elemente werden als Anwendungselement bezeichnet. (Nicht zu verwechseln mit XML-Elementen). Wenn nun ein Knoten im Baum gewählt wird, muss dieses Objekt an die Liste ‚**propagiert**' werden, damit diese den Inhalt des Objektes anzeigt. Die Anwendungselemente werden auf diese weise miteinander verknüpft.

1.1.4 Layout

Das Layout definiert die Aufteilung der gesamten Fläche des Tools. Diese kann horizontal oder vertikal in Teilflächen geteilt werden. Mit jeder dieser Teilflächen kann dies wiederum gemacht werden. Den resultierenden Flächen kann dann ein Anwendungselement zugeordnet werden.

1.1.5 Operation

Das System kennt Operationen, welche es ausführen kann. Diese Operationen können von Popup-, Menu und Drag&Drop angestossen werden.

1.2 Aufbau der Konfiguration

Der Aufbau der Konfiguration ist nachfolgend übersichtlich, dafür nicht vollständig wiedergegeben.

Tool (1)
- Layout (1)
- Application (1..n)
  - SubLayout (0..n)
  - Application Element (1..n)

Propagate (0..n)
PopupMenu (0-n)
Item (1..n)

Execute (0..n)
Operations (1)
Operation (0..n)
Toolbar (0..n)
ToolbarButton (0..n)
Execute (0..n)
Menubar (1)
Menu (0..n)
Item (0..n)

Execute (0..n)
PopupMenu (0-n)
Item (0..n)
Execute (0..n)
DragDrops (1)
DragDrop (0..n)
Execute (1..1)

In den Klammern () ist das minimale und maximale Vorkommnis angegeben.

1.3 Format und Konventionen der Konfiguration

Die Definition befindet sich in einem XML Dokument

Die XML-Elemente beginnen jeweils mit Grossbuchstaben.

Die XML-Attribute jeweils mit Kleinbuchstaben, mit Ausnahme von ID.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ToolConfiguration xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:noNamespaceSche-
maLocation="http://www.byron.ch/downloads/dokumente/BisToolConfiguration.xsd">
    <!-- hier folgt die Konfiguration -->
</ToolConfiguration>
```

Das Document-Element `<ToolConfiguration>` wird nachfolgend als Tool bezeichnet.

Für die Attributwerte wird jeweils ein Datentyp angegeben. Beispiele:

- `bw = "schwarz" ¦ "weiss"` definiert eine Aufzählung
- `size = <integer>` definiert eine Zahl
- `name = <text>` definiert einen Text
- `name = <boolean>` definiert einen Wahrheitswert
  `0` oder `false` für falsch/nein/aus
  `1` oder `true` für wahr/ja/ein
- `color = <color>` definiert eine Farbe der Form ‚#RRGGBB' (z.B. ‚#ff0000' für rot)

2 Tool

2.1 Hilfe

Die Hilfe ist ein Subelement des Tools und beinhaltet den Eintrag in der Help.hid Datei.

```
<Help>Raumbuch</Help>
```

Verweist auf chm::/Module/mod_exp_room.htm mit der Zeile

```
Raumbuch=Help.chm::/Module/mod_exp_room.htm
```

in der hid-Datei.

Spezifischere Hilfe-Einträge können durch <Help> in <Application> oder Application-Elements erreicht werden.

2.2 Dockzones (erst ab Byron/BIS v5.1.2)

Die Zonen, in denen dockbare Formulare angedockt werden können (links, unten und rechts), werden durch das optionale Element `Dockzones` konfiguriert:

```
<Dockzones disableAll="true" />
```

Aktuell ist nur das Attribut `disableAll` implementiert.

2.3 Einstellungen

Das Menü für die Einstellungen kann mit dem folgendem Subelement des Tools ausgeblendet werden:

```
<Settings enabled="false"/>
```

Als Default ist das Menü "Einstellungen" sichtbar, wenn das entsprechende Benutzergruppenrecht dies erlaubt.

Es kann konfiguriert werden, welches Einstellungsformular verwendet werden soll.

```
<Settings formName="MyToolForm"/>
```

Der Wert von "formName" entspricht der Bezeichnung der Formular-Objekte (bisB_FormObject). Wenn kein Wert angegeben wird, dann öffnet sich das Standardformular.

2.4 Übrige Definitionen

Das Tool definiert einen Layout und ein oder mehrere Application. Diese sind je Subelemente des Tools.

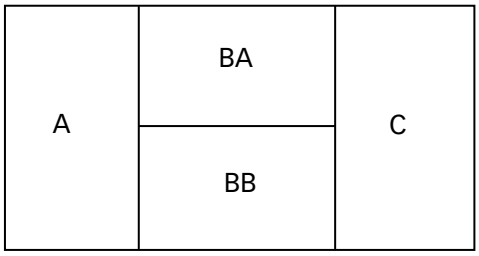
3 Layout / Panel

Das Layout definiert die Aufteilung der Fläche. Er beinhaltet Panel oder Layout. Das Layout kann beliebig verschachtelt werden. Ein Layout sollte immer mindestens 2 Elemente (Panel/Layout) enthalten. Ein Panel definiert schlussendlich die Fläche, d.h. ein Panel kann nichts mehr enthalten

Beispiel 1

```
<Layout>
  <Panel ID="A"/>
  <Layout>
    <Panel ID="BA"/>
    <Panel ID="BB"/>
  </Layout>
  <Panel ID="C"/>
</Layout>
```

definiert die Flächen A, BA, BB und C



3.1.1 XML-Attribute

Bezeichnung	Beschreibung
ID	`ID = <text>` Optional. ID des Layouts. Muss eindeutig sein Das Layout mit der (vordefinierten) ID `applicationControl` wird in die NavBar zum Umschalten von Applikationen installiert.
orientation	`orientation = "horizontal" ¦ "vertical"` definiert, ob die enthaltenen Panels/Layout horizontal oder vertikal angeordnet werden. Als Default gilt das Gegenteil des übergeordneten Layout, bzw. horizontal beim Obersten.
splitSize	`splitSize = <integer>` definiert den Abstand zwischen den Flächen in Pixel. Die so entstehenden Streifen kann der Benutzer zum ‚Verziehen' (resize) der Flächen verwenden. Als Default gilt der Wert des übergeordneten Layout, bzw. 5 beim Obersten.
splitLine	`splitLine = <boolean>` definiert ob eine gestrichelte Trennlinie gezeichnet werden soll. Als Default gilt der Wert des übergeordneten Layout, bzw. ‚false' beim Obersten.
size	`size = <integer> (erst ab v4.8.5)` definiert die initiale Grösse des Layouts in Prozent der Grösse des übergeordneten Layouts.
distribute	`distribute = <boolean> (erst ab v4.8.5)` Bestimmt, ob bei der initialen Grössenbestimmung die Flächen gleichmässig verteilt werden. Diese Eigenschaft wird an die darunterliegenden Layouts weitervererbt.

Bezeichnung	Beschreibung
noGradient	`noGradient = <boolean> (erst ab v4.10.4)` Ist dies gesetzt, so wird kein Farbverlauf als Hintergrund verwendet. Diese Eigenschaft wird an unterliegende Layouts vererbt.

3.2 Panel

Definiert eine Fläche innerhalb des Layouts. Panels werden von ApplicationElements belegt.

3.2.1 XML-Attribute

Bezeichnung	Beschreibung
ID	`ID = <text>` Mussfeld. Durch die ID wird das Panel von den Application-Elements identifiziert. Sie muss daher eindeutig sein.
minWidth	`minWidth = <integer>` Definiert die minimale Breite in Pixel, auf welche das Panel verkleinert werden kann.
minHeight	`minHeight = <integer>` Definiert die minimale Höhe in Pixel, auf welche das Panel verkleinert werden kann.
maxWidth	`maxWidth = <integer>` Definiert die maximale Breite in Pixel, auf welche das Panel vergrössert werden kann.
maxHeight	`maxHeight = <integer>` Definiert die maximale Höhe in Pixel, auf welche das Panel vergrössert werden kann
visible	`visible = <boolean>` Bestimmt die *initiale* Sichtbarkeit des Panels.
size	`size = <integer> (erst ab v4.8.5)` definiert die *initiale* Grösse des Panels in Prozent der Grösse des übergeordneten Layouts.

Beispiel 2

```
<Layout orientation="horizontal" splitLine="true">
  <Panel ID="A" size="20")/>
  <Layout" splitLine="false" splitSize="3" distribute="true">
    <Panel ID="BA"/>
    <Panel ID="BB"/>
  </Layout>
  <Panel ID="C"/>
  <Panel ID="D"/>
<Layout>
```

Die initiale Grösse eines Layouts (Breite oder Höhe je nach Orientation des übergeorndeten Layouts) wird folgendermassen bestimmt:

- Wenn `distribute="false"`, dann …
  - wenn `size` angegeben, dieses verwenden…
  - sonst, wenn entsprechendes `min…` (`minWidth`, `minHeight`) angegeben, dieses verwenden…
  - sonst, wenn entsprechendes `max…` (`maxWidth`, `maxHeight`) angegeben, dieses verwenden…
- sonst gleichmässig verteilen…

… und dann auf Constraints prüfen.

4 Application

4.1 XML-Attribute

Bezeichnung	Beschreibung
applicationID	`applicationID = <Text>` Identifiziert die referenzierte Applikation Erst ab Byron/BIS v5.5.1
classicPropagation	Legt fest, ob die Anwendung die bisherige Methode zum Erkennen von Propagate-Zyklen verwenden soll. Die neue Methode (`classicPropagation="false"`) ist etwas gutmütiger. Erst ab Byron/BIS v4.11.8 Vorgabe: `classicPropagation="true"`
defaultDropElement	`defaultDropElement = <Text>` Optional. Definiert das Drop-Ziel (ID des Application-Elements) für Drag&Drop auf die NavBar. Die Anwendung von defaultDropElement ist nur sinnvoll in Zusammenhang mit mehreren Applikationen in einem Werkzeug. Erst ab Byron/BIS v4.8.0.
focusedElement	`focusedElement = <text>` Optional. ID eines Applikationselements.
groupRight	`groupRight = <Text>` Optional. Definiert das benötigte Gruppenrecht für die Applikation. Die Anwendung von groupRight ist nur sinnvoll in Zusammenhang mit mehreren Applikationen in einem Werkzeug. Erst ab Byron/BIS v4.8.0. Ab Byron/BIS v4.10.8 können mehrere Gruppenrechte durch Komma getrennt angegeben werden. Das Recht ist vorhanden, wenn mindestens eines der angegebenen Gruppenrechte vorhanden ist.
icon	`icon = <text>` Optional. Name des Ikons, welches für die Applikation verwendet wird. Ab Byron/BIS v4.11.8: dem Icon können mit ‚+' Overlays hinzugefügt werden. Z.B. `icon="bisP_Order+state_new"`
ID	`ID = <text>` Optional. Bezeichnung für die Application. Wird u.a. von Load/SaveState verwendet. Default = Name des XML-Element ("Application").

storeWorkStateAfterPropagate	Vorgabe für das Attribut storeWorkStateAfterPropagate aller enthaltenen Application Elements. Wird storeWorkStateAfterPropagate="true" gesetzt, erfolgt ab Byron/BIS v5.0.0 eine Warnung, wenn nicht history="true". Erst ab Byron/BIS v4.11.0. Default = "false"
toolId	`toolId = <text>` Hat ab Byron/BIS v5.5.1 zwei Anwendungszwecke: 1. Setzt den Default der Tool-ID, d.h. dieser Wert wird zur Laufzeit vom Datenbankobjekt Tool überschrieben. Optional 2. Identifiziert das Tool von welchem eine referenzierte Applikation übernommen wird. `toolID` kann die ObjektID (Objekt-URL), die Doc_ID oder die bisB_ToolID des Tools enthalten. Erst ab Byron/BIS v5.5.1.
visible	`visible = <Boolean>` Optional. Definiert die Sichtbarkeit der Applikation. Mittels Setzen von `visible="false"` können einzelne Anwendungen einfach ausgeblendet werden. Erst ab Byron/BIS v4.10.6. Ab v4.11.8 kann `visible = "modal"` gesetzt werden, damit werden Modal Application definiert.
history	`history = <Boolean>` Optional. Definiert die Sichtbarkeit der History-Knöpfe (Vorwärts/Zurück). Die Verwendung von history="true" macht nur Sinn, wenn auch storeWorkStateAfterPropagate="true" an mindestens einem ApplicationElement der Application gesezt ist. Die Verwendung eines `FunctionControl` setzt implizit `history="true"`. Erst ab Byron/BIS v5.0.0. Default = "false"

4.2 Enthaltene XML-Elemente

Bezeichnung	Beschreibung
Help	Hilfe: Vgl. Tool damit wird die Hilfe des Tools überschrieben
Caption	Definiert die Bezeichnung (Titel) der Applikation, wie sie dem Benutzer präsentiert wird. Mehrsprachigkeit: siehe Caption
SubLayout	

Application Elements	
StatusBar	Definiert die Infozeile einer Applikation.
Operations	Das Operations Element beinhaltet alle Operation
Toolbar	Die Elemente Toolbar beinhalten alle ToolbarButton, Menu und ToolbarSeparator
ReferenceDate	Damit wird ein Control für den Stichtag eingeblendet.
Menubar	Das Menubar Element beinhaltet alle Menu und Item
PopupMenu	Das PopupMenu Element beinhaltet alle Menu und Item
DragDrops	Das DragDrops Element beinhaltet alle DragDrop Elemente
Print	Siehe Print-Element
Visibility	Bestimmt die Sichtbarkeit der Application, Siehe Visibility

4.3 SubLayout

Mit den Elementen < SubLayout > kann das (applikationsspezifische) Layout definiert werden.

4.3.1 Attribute von SubLayout

- `master = <Text>` definiert die (optionale) Bezeichung der Toolbar. Nur Toolbars mit Bezeichnungen können vom Benutzer im Kontextmenü ausgeblendet werden.
- Die weiteren erlaubten Attribute entsprechen denen des Elements Layout.

4.3.2 Elemente von SubLayout

- Die erlaubten Elemente entsprechen denen des Elements Layout.

Beispiel:

```
<SubLayout master="applicationContents" orientation="horizontal">
    <Panel ID="pnTree1" minWidth="100" minHeight="150"/>
    <Panel ID="pnGrid1" minWidth="300"/>
</SubLayout>

<SubLayout master="applicationControl" orientation="horizontal">
    <Panel ID="pnTreeC1" minWidth="100" minHeight="150"/>
    <Panel ID="pnGridC1" minWidth="300"/>
</SubLayout>
```

4.4 StatusBar

Mit dem Element <StatusBar> kann die Infozeile für eine Applikation konfiguriert werden. Es werden die Anzahl und Grösse der Felder (Panels) der Infozeile definiert.

Als Standard (ohne das Element <StatusBar>) enthält die Infozeile vier Felder mit den Grössen 20, 290, 170 und 50 Pixeln.

NB: das erste Feld der Infozeile mit einer Grösser von 20 Pixeln wird **immer** erzeugt.

Beispiel:

```
<StatusBar>
    <StatusPanel size="70"/>
    <StatusPanel size="50"/>
    <StatusPanel size="100"/>
    <StatusPanel size="200"/>
    <StatusPanel size="30"/>
</StatusBar>
```

4.5 Visibility

Erst ab Byron/BIS v4.10.8.

Die Sichtbarkeit der Application kann mittels einer FilterNavigation gesetzt werden. Wenn diese FilterNavigation ein Ergebnis liefert, ist die Application sichtbar. Als Startmenge wird das Dokument des Werkezugs mitgegeben.

Wichtig: Diese FilterNavigation wird nur einmalig beim Öffnen des Werkzeugs ausgeführt. Es können keine Tool-spezifischen Parameter verwendet werden!

Die Sichtbarkeit der Application wird in der folgenden Reihenfolge bestimmt:

1. Ist groupRight angegeben, wird das angegebene Gruppenrecht (ab Byron/BIS v4.10.8 können auch, durch Komma getrennt, mehrere Gruppenrechte angegeben werden) geprüft.
2. Ist die Application nach obigem Punkt sichtbar, wird das Attribut "visible" ausgewertet.
3. Ist die Application nach obigem Punkt sichtbar, wird das Element "Visibility" ausgewertet.

Diese Art der Konfiguration der Sichtbarkeit wird auch bei Menüs und Toolbars verwendet.

4.6 Modal Application

Modale Application (ab V 4.11.8) werden mit visible="modal" definiert. Modale Application werden am Schluss definiert. Sie sind unsichtbar und können mit propagates aktiviert werden. Wenn eine modale Application aktiv ist, verdeckt sie alle anderen. Die Aktionen in der Modal Application werden im Function Control nicht mit protokolliert.

Anwendungszweck: Modal Application können u.a. in Tools mit FunctionControl anstatt modalen Formularen eingesetzt werden.

4.6.1 Wie springe ich zu einer modalen Application?

Anwendung: Springen aus einem FunctionControl in eine modale Application:

```
…
        <FunctionGroup caption="Offerte" title="Offerte $Object_Display_Kontext$">
          …
            <Function caption="Planansicht" name="jmp_Plan" />
          …
        <FunctionGroup
…
      <Propagate operation="custom" name="jmp_Plan" toApplication="AreaPlan"
                 target="form" />
…
```

4.6.2 Wie kehre ich aus einer modalen Application zurück?

Anwendung: Rücksprung aus einer modalen Application mittels einer Toolbar (Speichern und zurück/Abbrechen):

…

```
    <Toolbar>
      <ToolbarButton icon="icCheck">
        <Execute name="SaveData" />
      </ToolbarButton>
      <ToolbarButton icon="grCancel">
        <Execute name="CancelReturn" />
      </ToolbarButton>
    </Toolbar>
```

…

```
      <Operation ID="CancelReturn" opCode="opWorkStateReload"
                 caption="Abbrechen" />
      <Operation ID="SaveData" opCode="opPropagate"
                 caption="Speichern und zurück" toElement="form" />
```

…

4.7 Referenzierte Applikationen (erst ab Byron/BIS v5.5.1)

Applikationen können aus anderen Tools übernommen und wiederverwendet werden. Dazu müssen am XML-Element «Application» die Attribute *applicationID* und *toolID* gesetzt sein.

Mit dem Wert von *toolID* wird das Tool bestimmt, von welchem die Applikation übernommen wird. Beginnt *toolID* mit einem «{«, dann wird die ObjektID (Objekt-URL) zum Suchen des Tools verwendet. In allen anderen Fällen wird zunächst nach der *Doc_ID* des Tools und dann nach der *bisB_ToolID* des Tools gesucht.

Mit dem Wert von *applicationID* wird die Applikation innerhalb des identifizierten Tools bestimmt.

Die referenzierte Applikation muss sich in das referenzierende Tool einpassen. Es ist darauf zu achten, dass die Bezeichnungen der verwendeten Layouts / Panels zueinander passen etc.

Beispiel:

```
    <Application ID="Personen"
         toolId="{8B2B3105-52DE-4430-934D-DFC5A4C6B86B}"
         applicationID="Personen" />
```

N.B.:

- die Aufstartgeschwindigkeit des Tools verschlechtert sich, da zusätzliche Daten geladen und interpretiert werden müssen.
- Das Werzeug welches referenziert, darf keine `Expl_Root`-Objekte haben, da sonst u.U. diese geladen werden.

5 Application Element

Die verschiedenen Application-Elemente besitzen verschiedene XML-Elemente (<Grid>, <Tree> etc.). In diesem Kapitel werden nur die allgemeinen (gemeinsamen) Eigenschaften erklärt.

5.1 Liste der Application Elements

Bezeichnung	Beschreibung
Tree	Objektbaum
Grid	Objektliste
Gantt	Ganttelement
Graphic	Grafikelement
Form	Formular
PropertyList	Eigenschaftsliste
Planner	Tages- und Wochen-Objektkalender
PlannerMonth	Monats-Objektkalender
Calendar	Datumsauswahlelement
Search	Objektsuchelement
Web	Webbrowserelement
ExternalControl	Element für exteren ActiveX-Steuerelemente
NavBar	Applikationsauswahlleiste
Tabbed	Registerelement
Monitor	Überwachung von GlobalNotifies
Chart	Chartelement
Toolbar	Seperate Toolbar

Alle Application Elements lassen sich durch eine Anzahl gemeinsamer XML-Attribute und –Elemente konfigurieren.

5.1.1 Allgemeine XML-Attribute

Allgemeine XML-Attribute eines Application Elements

Bezeichnung	Beschreibung
ID	`ID = <text>` Optional. ID des Application Elements. Muss eindeutig sein. Default = Name des XML-Element, z.B. "Grid" für <Grid> etc.
panel	`panel = <text>` Mussfeld. Name des Panels, auf welchem das Application Element erscheinen soll.
onDoubleClick	`onDoubleClick = <text>` Optional. ID der Operation, welche beim Doppelklick auf das Application Element ausgeführt wird. Typischerweise also ‚öffnen', d.h. eine Operation mit `opCode="opOpen"` Wird **vor** `onDoubleClickPropagate` ausgeführt.

onDoubleClickPropagate	`onDoubleClickPropagate = <text>`
	Optional. Name des Propagates, welches bei einem Doppelklick auf das Application Element ausgeführt wird. Mit Hilfe mehrerer `<Propagates>` können so mehrere Operationen ausgeführt werden. Wird **nach** `onDoubleClick` ausgeführt.
	Beispiel:
	`<Grid ID="grid2" onDoubleClickPropagate="ShowDetails" />`
	Erst ab Byron/BIS v4.11.9

checkDoubleClickHandled	`checkDoubleClickHandled = <boolean>` Optional. Legt fest, ob bei einem Doppelklick das Eingangsapplikationselement (Applikationselement mit einem "load"-propagate auf das Element) zuerst prüfen soll, ob der Doppelklick ausgeführt werden soll oder nicht. z.B. wenn das Objekt dargestellt werden kann. Default = "true".
containerSupport	`containerSupport = <boolean>` Mit containerSupport werden Objekte in den Container erzeugt. Dies ist Default beim Grid. Ohne containerSupport werden die Objekte in das selektierte Objekte kopiert.
receiveSelectNotification	`receiveSelectNotification = <boolean>` Optional. Selektionen werden in BIS als ‚broadcast' verschickt (zusätzlich zu ‚propagate select'). Wenn ein Application Element diese Selektion ignorieren soll, dann muss selectNotification auf false gesetzt werden. Default = "true".
sendSelectNotification	`sendSelectNotification = <boolean>` Optional. Wenn dies gesetzt ist, dann sendet das Application Element (zusätzlich zu ‚propagate select') einen Selection-Broadcast. Dies bedeutet, dass andere Tools auf diese Selektion reagieren können. Default = "true".
actualizeWhenInactive	`actualizeWhenInactive = <boolean>` Optional. Normalerweise reagieren Applikationelemente nur auf ChangeNotifications und Aktualisierungen (F5), wenn sie Aktiv sind – unter Anderem wenn die zugehörige Applikation Aktiv (angezeigt) ist. Ist `actualizeWhenInactive` gesetzt, dann werden ChangeNotifications und Aktualisierungen (F5) immer verarbeitet. Default = "false". Erst ab Byron/BIS v5.1.2
optional	`optional = <boolean>` Optional. Wenn dies gesetzt ist, dann kann das Application Element über einen Schliess-Button versteckt werden. Hinweis: Das Verstecken und wieder sichtbar machen des Application Elements sollte auch über eine entsprechende

	Operation `opVisibleOnOff` ermöglicht werden. Default = "false".

toolId	`toolId = <text>`
	`Damit kann die Tool-Id der Applikation überschrieben werden. Dies kann zB. in einem Grid sinnvoll sein, damit für verschiedene Grids auch verschiedene Ansichten möglich sind.`
	`Vgl. opEditGridView, opNewGridView`

headerVisible	`headerVisible = <boolean>` Optional. Durch setzen auf "false" kann die Kopfzeile eines Application Elements versteckt werden. Default = "true".
propagateOnFocusChanged	`propagateOnFocusChanged = <boolean>` Optional. Wenn dies gesetzt ist, dann sendet das Application Element zusätzlich beim Fokuswechel eine Selektionsänderung (propagate). Dies bedeutet, dass andere Applications Elemente auch auf einen Focuswechsel reagieren können. Default = "false".
propagateNoObject	`propagateNoObject = <boolean>` Optional. Wenn dies gesetzt ist, dann sendet das Application Element eine Selektionsänderung (propagate) auch für eine leere Selektion. Default = "true".
deselectIfObjectNotPresent	`deselectIfObjectNotPresent = <boolean>` Optional. Wenn dies gesetzt ist, dann empfängt das Application Element eine Selektionsänderung (propagate) auch für nicht sichtbare Objekte. D.h. es wird ein Deselect durchgeführt, weil die Objekte nicht sichtbar sind. Wenn z.B. im Grid bei gefilterten Ansichten die Selektion nicht verloren gehen soll, dann muss der Parameter beim Empfänger-Element auf false gesetzt werden. Falls leere Objekte die Selektion auch nicht verändern sollen, dann muss zusätzlich der Parameter "propagateNoObject" am Sender-Element auf false gesetzt werden. Default = "true".
minimizable	`minimizable = <boolean>` Optional. Wenn dies gesetzt ist, kann das Application Element über einen Minimier- bzw- Maximierbutton minimiert oder maximiert werden. Default = "false"
minimized	`minimized = <boolean>` Optional. Wenn dies gesetzt ist, wird das Application Element minimiert gestartet. Default = "false"

groupCaption	`groupCaption =" <text>` Optional. Wenn diese Caption gesetzt ist, wird um das Application Element eine Gruppe mit der angegebenen Caption gezeichnet. Erst ab Byron/BIS v4.10.4
noBorder	`noBorder = <boolean>` Optional. Wenn dies gesetzt ist, wird um das Application Element kein Border gezeichnet. Dies wird im Moment nur vom Application Element Tree unterstützt. Default = "false" Erst ab Byron/BIS v4.10.4
parentColor	`parentColor = <boolean>` Optional. Wenn dies gesetzt ist, wird die Farbe des übergeordneten Layouts übernommen. Dies wird im Moment nur vom Application Element Tree unterstützt. Default = "false" Erst ab Byron/BIS v4.10.4
statusbarVisible	`statusbarVisible = <boolean>` Optional. Wenn dies gesetzt ist, wird unterhalb des Application Elements eine Statusbar angezeigt. Alle Statustexte werden in diese Statusbar ausgebeben. Ohne Angabe des Elements <StatusBar> im Application Element wird die Standard Statusbar verwendet. Default = "false". Erst ab Byron/BIS v4.10.4
storeWorkStateAfterPropagate	Gibt an, ob nach einem Propagate ein Application Work State aufgenommen werden soll. Kann auch an der Application konfiguriert werden. Siehe auch opWorkStateRecord. Erst ab Byron/BIS v4.11.0. Default = "false" bzw. der Wert von storeWorkStateAfterPropagate der Application.

5.1.2 Allgemeine XML-Elemente

Allgemeine XML-Elemente eines Application Elements

Bezeichnung	Beschreibung
Caption	Legt die Beschriftung des ApplicationElements fest.
Help	Damit wird die Hilfe von <Application> überschrieben
Propagate	Definiert das Propagieren der Selektion zu anderen Application Elements

StatusMapping	Definiert die Verwendung Felder der Infozeile durch das Application Element.
PopupMenu	Definiert die Erweiterungen des PopupMenüs für das Application Element.
Toolbar	Definiert die Toolbar für das Application Element.
Contents	Definiert welche Objekte das Application Element zeigt
ContentsInverse	
StartObjects	Definiert, welche Objekte von `<Contents>` als Ausgangspunkt verwendet werden.

5.2 Caption

Der Inhalt von Caption wird zur Beschriftung verwendet. Die Caption kann als Darstellungsformel (eingeschlossen durch $-Zeichen) und/oder übersetzbarer Text (eingeschlossen durch @-Zeichen) verwendet werden.

Beispiele:

```
<Caption>Raumbuch</Caption>
```

```
<Caption>@sMyList@</Caption>
```

```
<Caption>Raum: $Object_Diplay_Kontext$</Caption>
```

5.3 Toolbar (erst ab Byron/BIS v4.8.9)

Toolbars werden unterhalb des Header platziert. Es sind beliebig viele Toolbars innerhalb eines Application Elements erlaubt. Die einzelnen Toolbars können jedoch nicht verschoben und ausgeblendet werden.

Definition der Toolbar siehe Toolbar

5.4 Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
OBJ_Tool	Objekt	Das Werkzeug
VAL_ToolId	Text	Attribut bisB_ToolId von Tool
VAL_ApplicationId	Text	ID der aktuellen Application (ab v4.9.10)
VAL_FocusedElementId	Text	Die ID des fokusierten Elements
VAL_IsPropagating	Boolean	Gibt an, ob das ApplicationElement zur Zeit am Propagieren ist (ab v5.0.0)
OBJ_StartObjects	Objekt	Startobjekte des Applikationselements
OBJ_Container	Objekt	Container des Applikationselements
OBJ_Selection	Objekt	Selektion des Applikationselements

Der Parameter muss die Element-ID als Prefix beinhalten.

```
<Condition> RECALL "GridRooms.OBJ_Selection" CLASS "Room"</Condition>
```

Weitere Werte sind je nach Operation bzw. Applikations-Element möglich. zB. VAL_CurrentDate im Planner oder VAL_TargetElementId in Drag&Drop Operationen.

5.5 Contents / Contents Inverse

Contents definiert die Navigation vom Eingangsobjekt zu den darzustellenden Objekten.

Beispiel:

```
<Contents>[VIA Object_To_Children OR VIA Room_To_Component]</Contents>
```

Das Eingangsobjekt wird durch ein Propagate operation=load oder durch `<StartObjects>` geliefert. Wenn das Application Element von keinem propagate als target referenziert wird, dann muss die Contents Navigation auch ohne Eingangsobjekt etwas liefern. Dazu sind in der Filternavigation Parameter definiert, welche verwendet werden können, zum Beispiel das Werkzeug als 'Tool'.

Beispiel:

```
<Contents> DATAOF 'Tool' VIA Expl_Root CLASS Space</Contents>
```

Liefert das Space-Objekt, welches mit dem Tool verknüpft ist.

Häufig möchte man eine Kombination dieser Möglichkeiten, dies lässt sich mit einem ELSEIF=0 realisieren:

```
[
    CLASS Space
ELSIF = 0
    DATAOF 'Tool' VIA Expl_Root
ELSIF = 0
    START INSTANCES Root_Space
]
```

- direkt den Eingang: mit propagate vorgeschaltet Verknüpfungsleiste
- Mit dem Werkzeug assoziiertes Space-Objekt
- Wurzel Objekt der Space-Objekte.

Optional kann mit Contents Inverse die Navigation von den darzustellenden Objekten zu den übergeordneten Objekten definiert werden. Die inverse Contents Navigation wird für das Zeigen von Objekten benötigt, um das Objekt in der Liste zu finden und das übergeordnete Objekt im Baum zu selektieren. Wird keine Contents Inverse Navigation angegeben, versucht das System die Contents Navigation automatisch zu invertieren. Dies funktioniert nicht bei einseitigen (virtuellen) Assoziationen.

Beispiel:

```
<ContentsInverse>
    [
        VIA Object_To_Parent
    OR
        VIA Room_Of_Component
    ]
</ContentsInverse>
```

5.6 StartObjects

StartObjects definiert eine Navigation, mit welcher alternativ Eingangsobjekte ermittelt werden können. Diese Eingangsobjekte werden dann von `<Contents>` als Startmenge verwendet.

Beispiel:

```
<StartObjects>START USER VIA bisB_UserToList</StartObjects>
```

Für die durch `<StartObjects>` definierte Navigation gelten dieselben vordefinierten Parameter wie für `<Contents>`.

N.B. Es ist für die **Performance** der Tools günstig, wenn die StartObjects-Navigation mit einem Klassenfilter abgeschlossen wird.

5.7 Propagate

Wenn der Benutzer in einem Application Element ein Objekt selektiert, dann kann dieses Objekt mit `<Propagate>` an ein anderes Application Element weiter gegeben werden.

Wie in einem dynamischen Formular propagates ausgelöst, bzw abgefangen werden:
Propagate in Formularen

Zyklen in den Propagates verschiedener Applikationelemente werden entdeckt und beenden das Propagieren. Ab Byron/BIS v4.11.8 kann die Methoden zur Erkennung eines Zyklus gewählt werden. Vgl. Application (classicPropagation).

5.7.1 XML-Attribute

Bezeichnung	Beschreibung
andUp	`andUp = <boolean>` Optional. Wenn zu einem Element (oder einer Operation) eines anderen Tools propagiert wird, dann wird mit diesem Parameter gesteuert, ob das andere Tool in den Vordergrund gebracht wird. Vorgabe = false. Erst ab Byron/BIS v4.11.8.
codeTarget	`codeTarget = <integer>` Optional. Setzt den Propagate-Code für den Empfänger.
name	`name = <text>` Optional. Name wird als Filter verwendet. D.h. es werde nur Propagates ausgeführt, bei denen der Name übereinstimmt.
nameTarget	`nameTarget = <text>` Optional. Setzt den Propagate-Namen für den Empfänger.

	Ab Byron/BIS v5.2.1 kann nameTarget auch eine Expression enthalten. Eine Expression muss mit einem ‚=' beginnen. Beispiel: `nameTarget="=IF (COUNT < 2) THEN 'showOne' ELSE 'showMany'"`
oneStep	`oneStep = <boolean>` Optional. Führt das Propagate nur für eine Stufe aus, d.h. das propagierte Element propagiert selber nicht weiter. Vorgabe = false
operation	`operation = "load" | "select" | "custom" | "show" | "execute"` Mussfeld. Definiert das Verhalten für das unter `target` angegebene Application Element: `load`: definiert das darzustellende Objekt `select`: selektiert das Objekt, sofern vorhanden `custom`: spezifische Operation des target Element (beim jeweiligen Application Element dokumentiert) `show`: Show-Operation an target Element `execute`: Führt die durch target bezeichnete Operation aus. Die Objekte für die Operation werden mit Hilfe von `forElement` der Operation ermittelt. Erst ab Byron/BIS v4.11.8.
parameterTarget	`parameterTarget = <text>` Optional. Setzt den Parameter für den Empfänger. Siehe auch fromParameter bei opPropagate oder Function. Erst ab Byron/BIS v4.11.8.
showWarnings	`showWarnings = <boolean>` Optional. Vorgabe = true. Bei Pages: Gibt an ob Warnungen aufgrund der Konfiguration in das Log geschrieben werden sollen. Erst ab Byron/BIS v4.11.0 Die Fehlermeldung Application mit ID="XY" nicht gefunden kann mit showWarnings="false" unterdrückt werden. Dies kann sinnvoll sein, wenn die Application nicht für alle Benutzer sichtbar ist. Erst ab Byron/BIS V5.2.1
toApplication	`toApplication = <text>` Optional – gibt an, in welcher Anwendung sich das `target` befindet. **N.B.** Wird an eine andere Anwendung propagiert, dann wird diese Anendung aktiviert. Vorgabe: *leer* = die eigene Anwendung. Vergleiche auch showWarnings.

target	`target = <text>` Mussfeld. Entweder die ID des Application Elements, an welches die selektierten Objekte übergeben werden sollen (`load .. show`) oder die ID der Operation, welche durch `execute` ausgeführt wird.

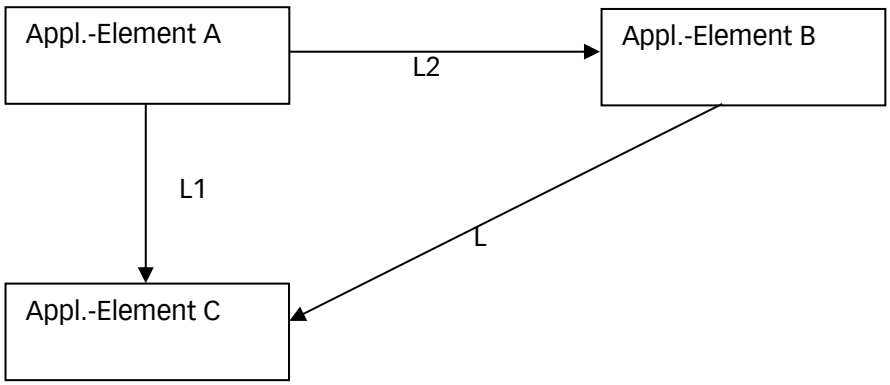
5.7.2 XML-Elemente

Bezeichnung	Beschreibung
Condition	Mit der Condition-Filternavigation kann das Ausführen von Propagate verhindert werden. Wenn kein Condition-Element existiert, wird Propagate immer ausgeführt. Als Eingangsobjekte (Input) werden die Propagate-Objekte verwendet.
Tool	Mit der Tool-Filternavigation kann das Tool ermittelt werden, an welches die Objekte propagiert werden sollen.
Transform	Mit der Transform-Filternavigation können die weitergegebenen Objekte festgelegt werden. Als Eingangsobjekte (Input) werden die Propagate-Objekte verwendet.

5.7.3 Beispiele:

```
<Propagate operation="load" target="form"/>

<Propagate operation="load" target="grid">
    <Condition>CLASS Space</Condition>
    <Transform>VIA Room_To_Component</Transform>
</Propagate>

<Propagate operation="custom" name="container" target="dockedForms"/>

<Propagate operation="load" target="form" nameTarget="A" codeTarget="1"/>
```

Die Propagate werden in der spezifizierten Reihenfolge ausgeführt. Wenn ein Element ein Propagate erhalten hat, werden alle weiteren desselben Types ignoriert. Das auslösende Application Element ignoriert empfangenen propagate.

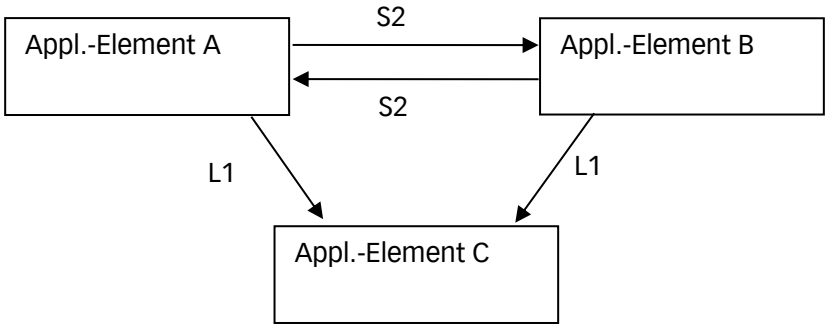


-

L = Propagate operation = "load"

S = Propagate operation = "select"

- Die Nummer gibt die Reihenfolge in der XML Definition an. In diesem Beispiel wird im Application Element A zuerst das load nach ‚C' deklariert und danach das load nach ‚B'.
- Wird im Application Element A ein Objekt selektiert, dann geschieht folgendes:
- A –L1-> C Appl.-Element C wird geladen.
  A –L2-> B Appl.-Element B wird geladen
  B –L-> C Appl.-Element C wird NICHT geladen, das es bereits geladen wurde.
- Würde man L1 und L2 vertauschen, ergibt sich ein anderer Ablauf:
- A –L1-> B Appl.-Element B wird geladen.
  B –L-> C Appl.-Element C wird geladen
  A –L2-> C Appl.-Element C wird NICHT geladen, das es bereits geladen wurde.
- Der Anwender Erwartet das gewählte Objekt im Appl.-Element C, im allgemeinen wird dies aber nicht geschehen, da das Appl.-Element B durch das Laden alles deselektiert und 'kein-Objekt' an C propagiert..
- Faustregel: Die direkten 'Loads' vor den indirekten.



-

L = Propagate operation = "load"

- S = Propagate operation = "select"

- Die Nummer gibt die Reihenfolge in der XML Definition an, d.h. hier steht das L(oad) vor dem S(elect). Wenn in diesem Beispiel im Application Element A ein Objekt Selektiert wird, dann geschieht folgendes:
- A –L1-> C Appl.-Element C wird geladen.
  A –S2-> B in Appl.-Element B wird selektiert.
  B –L1-> C Appl.-Element C wird NICHT geladen, da es bereits geladen wurde.
- Würde die Reihenfolge von L und S vertauscht, dann würde folgendes passieren:
- A –S1-> B in Appl.-Element B wird selektiert.
  B –L1-> C Appl.-Element C wird geladen.
  A –L2-> C Appl.-Element C wird NICHT geladen, da es bereits geladen wurde.
- Dies kann für den Anwender ein unerwartetes Verhalten zurfolge haben. Wenn das in A selektierte Objekt in B nicht vorkommt, dann wird in C kein Objekt dargestellt. N.B. auch 'kein Objekt' wird propagiert.
- Faustregel: Load vor Select

5.8 Selection

Mit Selection kann die native Selektion des Application Elements geändert werden.

```
<Selection>
    [OR VIA Room_To_Component]
</Selection>
```

Als Eingang der Filternavigation kommt das selektierte Objekt die resultierenden Objekte sind dann die effektive Selektion. Mit der Operation `opNavParameter` kann dieses Verhalten auch vom Benutzer gesteuert werden.

Beispiel mit:

```
<Operation ID="NavParameterA" opCode="opNavParameter"
           caption="Parameter A" paramName="nav_param_a"
           uncheckedValue="1" checkedValue="2">

<Selection>[OR CONDITION "= nav_param_a = 2" VIA Room_To_Component]</Selection>
```

5.9 Empfangene propagates

Folgende Custom-Propagates sind definiert:

Name (`propagateName`)	Beschreibung
caption	Setzt die Caption zur Laufzeit. Beispiel: `<Propagate operation="custom" target="grid" nameTarget="caption" parameterTarget="'Dies überschreibt Caption'"/>` Die übergebenen Texte werden übersetzt. TextIDs können mit @ angegeben werden: @FloorSearch@. Siehe auch `<Caption>` oben. Erst ab Byron/BIS v4.11.8.

N.B. Der Wert von *propagateName* wird case insensitive ausgewertet.

5.10 StatusMapping

Mit dem Element Statusmapping wird die Zustandsausgabe des Application Elements den Feldern (Panels) der Infozeile zugewiesen. Das `<StatusMapping>` enthält nur Elemente mit der Bezeichnung `<StatusMapEntry>`

5.10.1 StatusMapEntry

Ein StatusMapEntry weist eine (1) Zustandsausgabe eines Application Elements einem (1) Panel der Infozeile zu.

XML-Attribute

Bezeichnung	Beschreibung
ID	`ID = <text>` Mussfeld. Identifiziert die Zustandsausgabe eines Application Elements. Diese IDs werden durch die Application Elements definiert und sind in den entsprechenden Kapiteln dokumentiert. Beispiele: GridContents, GridAutoSum, GridPreferredView.
panel	`panel = <integer>` Mussfeld . Nummer des Status-Panels, auf welchem der Application Element-Status angezeigt werden soll. Beginnt mit 1.

Beispiel

```
<StatusMapping>
    <StatusMapEntry ID="GridContents" panel="1"/>
    <StatusMapEntry ID="GridAutoSum" panel="2"/>
    <StatusMapEntry ID="GridPreferredView" panel="3"/>
</StatusMapping>
```

6 Tree

Mit dem Application Element <Tree> können Objekte in einer Hierarchie dargestellt werden.

Zusätzliche XML-Attribute

Bezeichnung	Beschreibung
history	`history = <boolean>` Mit history="true" wird das Application Element angewiesen sich den Verlauf zu merken. Der gespeicherte Verlauf kann durch die Operationen `opTreeHistoryBackward` und `opTreeHistoryForward` verwendet werden.
display	`display = <text>` Enthält das Byron/BIS Attribut, welches für die Darstellung verwendet wird. Erst ab Byron/BIS v4.7.5 siehe auch Element `<Display>`
sort	`sort = <text>` Enthält das Byron/BIS Attribut, welches für die Sortierung verwendet wird. siehe auch Element `<Display>`
sortAsc	`sortAsc = <boolean>` Gibt die Sortierreihenfolge für das unter `sort` angegebene Byron/BIS Attribut an: "true" = aufsteigend (Default) "false" = absteigend siehe auch Element `<Display>`
expanded	`expanded = <integer>` Gibt die Anzahl der Stufen an, die beim ersten Laden des Baumes geöffnet werden sollen, an. Default = "0"

expandedLoad	`expandedLoad = <integer>`
	`Gibt die Anzahl der Stufen an, die beim jedem Laden des Baumes geöffnet werden sollen, an.`
	`Default = "0"`

fullRefreshOnActualize	`fullRefreshOnActualize = <boolean>` Wenn "true", aktualisiert der Baum bei einem Actualize (F5) auch seine Knoten-Hierarchie. Wenn "false", wird nur aktualisiert wenn eine ChangeNotification eintrifft (nicht F5) und dabei die Menge der StartObjekte verändert wurde. Änderungen von Attributen werden dabei aber nicht berücksichtigt (nur Assoziationen, Creates, etc.) Default = "false" Erst ab Byron/BIS v4.8.6 - Vorher Verhalten wie bei "false"

checkIfContentsChangedOn-Notf	`checkIfContentsChangedOnNotf = <boolean>` Wenn "true", werden nicht nur StartObjects angeschaut, ob sich etwas geändert hat, sondern auch gecheckt ob das Ergebnis der Contents-FilterNavigation unterschiedlich ist und nur dann ein Reload des Trees gemacht. Default = "false" Erst ab Byron/BIS v4.9.1 - Vorher Verhalten wie bei "false"

Zusätzliche XML-Elemente

Element	Beschreibung

Display (ab v4.9.12)	Damit können die Attribute `display` (durch `attribute`), `sort` und `sortAsc` vom `<Tree>` überschrieben werden. Display wird nur angewendet, wenn die enthaltene Filternavigation das Objekt durchlässt.
	Attribute:
	• `attribute = <text>`
	• `sort = <text>`
	• `sortAsc = <boolean>`
	`<Display attribute="Object_Display_Kontext" sort="bisB_Sort" sortAsc="false">`
	`CLASS Space`
	`</Display>`

SubContentsExists (ab v4.9.0)	Enthält eine Filternavigation, die steuert, ob vor dem Knoten ein "+" gezeichnet werden soll (ist das Resultat leer -> kein "+", ist das Resultat nicht leer -> "+"). Ist SubContentsExists nicht vorhanden, wird SubContents mit derselben Semantik verwendet.
	`<SubContentsExists>`
	`[ CLASS JahresOrdner`
	`CONDITION "=OBJCOUNT(,JahresOrdnerToMessung') > 0"`
	`OR`
	`CLASS Station`
	`CONDITION "=OBJCOUNT(,Object_To_Children') > 0"`
	`]`
	`</SubContentsExists>`

Zusätzliche Beschreibungen zu den erhaltenen Propagates

Propagate-Typ	Beschreibung
select	Klappt den Baum auf und zeigt das propagierte Objekte im Baum an (wie die Show-Operation auf dem Tree). Ist der propagate Code = 1, wird die Selektion vom Tree nicht weiter propagiert. `<Propagate operation="select" target="treeSpaces" codeTarget="1"/>`

Beispiel

```
<Tree ID="tree" panel="pnTree" history="true" display="Name" sort="Object_Display_Kontext"
optional="true">
    <Caption auto="false">@sContainer@</Caption>
    <Contents>
        [
            CLASS Doc_Container ELSIF = 0
            RECALL 'Tool' VIA Expl_Root ELSIF = 0
            START CLASS Doc_Root_Container
        ]
    </Contents>
    <SubContents>
        VIA Object_To_Children CLASS Doc_Container
    </SubContents>
    <SubContentsExists>
        CONDITION "=OBJCOUNT(‘Object_To_Children’) > 0"
        <!-- diese Filternavigation liefert da gewünschte Ergebnis nicht zu 100%
              da auch + vor Container gemalt wird, die keine Container enthalten -->
    </SubContentsExists>
    <Propagate operation="load" target="grid"/>
    <Propagate operation="custom" name="container" target="dockedForms"/>
    <PopupMenu>
        <Item positionAfter="PopupDelete"><Execute name="TreeSearch"/></Item>
        <Item positionAfter="PopupDelete" caption="-"/>
    </PopupMenu>
</Tree>
```

6.1 Implementierung StatusMapping

Das Application Element Tree implementiert eine Ausgabe auf den Statusbar (vgl. StatusMapping). Durch ein StatusMapEntry mit der ID `TreeSelection` wird der Text des selektierten Elements im der Statusbar angezeigt.

Beispiel

```
<StatusMapping>
    <StatusMapEntry ID="TreeSelection" panel="1"></StatusMapEntry>
</StatusMapping>
```

6.2 Tree-Operationen

6.2.1 opTreeHistoryForward

Diese Opertation ermöglicht das Navigieren zum nächsten gemerkten Knoten im Baum.

In den Explorern wird diese Funktion meistens im Menü "Ansicht" → "Gehe zu" → "Vorwärts" verwendet.

Beispiel

```
<Operation ID="TreeForward" opCode="opTreeHistoryForward" onElement="tree"/>
```

6.2.2 opTreeHistoryBackward

Diese Opertation ermöglicht das Navigieren zum vorherigen gemerkten Knoten im Baum.

In den Explorern wird diese Funktion meistens im Menü "Ansicht" → "Gehe zu" → "Zurück" verwendet.

Beispiel

```
<Operation ID="TreeBackward" opCode="opTreeHistoryBackward" onElement="tree"/>
```

6.2.3 opTreeGotoParent

Diese Opertation ermöglicht das Navigieren zum übergeordnetem Knoten im Baum.

In den Explorern wird diese Funktion meistens im Menü "Ansicht" → "Gehe zu" → "Übergeordnetem Element" verwendet.

Beispiel

```
<Operation ID="TreeParent" opCode="opTreeGotoParent" onElement="tree"/>
```

7 Grid

Mit dem Application Element <Grid> können Objekte in einer tabellarisch dargestellt werden.

Beispiel

```
<Grid ID="grid" panel="pnGrid" onDoubleClick="Open" containerSupport="true"
      dropShowTryOthersFirst="false">
    <Caption>@sContents@ "$Object_Display_Kontext$"</Caption>
    <Contents>
        VIA Object_To_Children CLASS Doc_Classes
    </Contents>
    <SubContents>
        [VIA Object_To_Children ] SORT (+Object_Display_Kontext)
    </SubContents>
    <GridView userinterface="true" includeStandard="true" alwaysPrefer="true">
        START INSTANCES GridView
    </GridView>
    <StatusMapping>
        <StatusMapEntry ID="GridContents" panel="1"/>
        <StatusMapEntry ID="GridAutoSum" panel="2"/>
        <StatusMapEntry ID="GridPreferredView" panel="3"/>
    </StatusMapping>
</Grid>
```

7.1 Attribute

Bezeichnung	Beschreibung
dropShowTryOthersFirst	`dropShowTryOthersFirst = <boolean> (erst ab v4.10.2)` Definiert, ob beim Zeigen (`opShow`) von Elementen die Input und Outputs vor den eigentlichen Objekten durchsucht werden. Dies ist insbesondere bei der Anwendung von QuickSearch sinnvoll um die Suchergebnisse im "richtigen Pfad" zu zeigen. Default = "false"
maxAutoFilterItems	`maxAutoFilterItems = <integer> (erst ab v4.10.3)` Definiert die maximale Anzahl der AutoFilter-Werte in der Auto-Filter-Combobox. Kann mit dem globalen Parameter `P_MaxAutoFilterItems` für alle Grids (auch in Formularen) eingestellt werden. Default = 1000

7.2 Element: GridView

Das Element <GridView> definiert durch eine Filternavigation die Ansicht-Objekte (Spalten) für das Grid. Dabei kann über das Attribut `includeStandard` gesteuert werden, ob die Standard-Ansicht zur Verfügung steht oder nicht. Mit dem Attribut `userinterface` kann die interaktive Auswahl und Anzeige der Ansichten ein- bzw. ausgeblendet werden.

Wird das Element <GridView> weggelassen, wird das Standardverhalten der Ansichten verwendet. D.h. Ansichten können interaktiv ausgewählt und angezeigt werden. Die Zuordnung bzw. Filterung der Ansichten erfolgt über das BIS-Attribut "Tool-ID" des Explorers.

Über das Attribut `alwaysPrefer` kann gesteuert werden, ob der Wechsel auf eine favorisierte Ansicht immer, oder nur wenn bereits eine favorisierte Ansicht gewählt ist, erfolgt.

Beispiel (Definition einer "festen" Ansicht ohne Auswahlmöglichkeiten durch den Benutzer)

```
<GridView userinterface="false" includeStandard="false">
    START INSTANCES GridView
    VALUE bisB_ToolId = DATAOF VAL_ToolId
    VALUE Name = ‘Personen’
</GridView>
```

Neue Elemente, Attribute und Schreibweise ab v4.11.8

```
<GridView userinterface="true" includeStandard="true" allowViewDelete="true"
          allowViewNew="true" allowViewEdit="true" restoreFromRegistry="true">
  <Default>
    START INSTANCES GridView VALUE Name = 'Mitarbeiter'
  </Default>
  <Contents>
     START INSTANCES GridView
  </Contents>
</GridView>
```

7.3 Element: SubContents

Erste Version ab **v4.10.8**. Das Element SubContents ermöglicht es, wie beim Tree, den Zeilen untergeordnete Elemente darzustellen.

```
<SubContents>
    VIA Object_To_Children
    SORT (+Object_Display_Kontext)
</SubContents>
```

7.4 Implementierung StatusMapping

Das Application Element Grid implementiert mehrere Ausgaben auf den Statusbar (vgl. StatusMapping).

Bezeichnung	Beschreibung
GridContents	Zeigt den Grid-Inhalt (Anzahl Objekte, selektiert etc.) in der Infozeile an.
GridAutoSum	Zeigt bei Mehrfachselektion die Summe der aktuellen Spalte in der Statusbar an.
GridPreferredView	Zeigt "Favorisierte Ansicht" in der Statusbar an, wenn die Ansicht favorisiert ist.

Beispiel (entspricht der Vorgabe):

```
<StatusMapping>
    <StatusMapEntry ID="GridContents" panel="1"/>
    <StatusMapEntry ID="GridAutoSum" panel="2"/>
    <StatusMapEntry ID="GridPreferredView" panel="3"/>
</StatusMapping>
```

7.5 Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
OBJ_CurrentView	Objekt	Aktuelle Ansicht
VAL_CurrentView	Text	Bezeichnung der aktuellen Ansicht.

Ab Version 5.1.2

7.6 Empfangene Propagates

Bezeichnung	Beschreibung
ResetAutoFilter	Setzt den Wert aller Auto-Filter zurück auf "(alle)" (erst ab Byron/BIS v4.11.5).
SetViews	Setzt die Menge der auswählbaren Ansichten (erst ab Byron/BIS v4.11.8). Beispiel: `<Propagate operation="custom" name="setViews" target="grid" /> <Transform>START INSTANCES GridView VALUE Name ~= "A*"</Transform> </Propagate>`
SetView	Setzt die Ansicht (erst ab Byron/BIS v4.11.8). **Achtung Falle**: Werden favorisierte Ansichten verwendet, dann können diese durch das Grid geändert werden. Ab Version 5.1.2 kann parameterTarget leer gelassen werden und die Ansicht auch als Objekt propagiert werden. Beispiel: `<Propagate operation="custom" name="setView" target="grid" parameterTarget="'Neue Ansicht'"/>`

7.7 Grid-Operationen

7.7.1 opGridTrackScroll

Mit dieser Operation kann der aktive Bildlauf (Ziehen der Bildlaufleisten führt den Inhalt der Liste sofort nach) für das Grid ein- beziehungsweise ausgeschaltet werden.

Beispiel

```
<Operation ID="TrackScroll" opCode="opGridTrackScroll"
           caption="Aktiver Bildlauf"/>
```

7.7.2 opGridOptimizeRowHeight

Mit dieser Operation kann die automatische Berechnung der optimalen Zeilenhöhe bei mehrzeiligen Texten ein- beziehungsweise ausgeschaltet werden.

Beispiel

```
<Operation ID="RowHeight" opCode="opGridOptimizeRowHeight"
           caption="Optimale Zeilenhöhe"/>
```

7.7.3 opEditGridView

Die Operation EditGridView öffnet ein Dialogfeld zur Bearbeitung der aktuellen Ansicht des zugeordneten Grid-Elements. Wenn forGrid weggelassen wird, dann wird das fokusierte Grid genommen.

Die Operation kann in Menüs, PopupMenüs und ToolbarButtons verwendet werden.

Beispiel:

```
<Operation ID="ViewEdit" opCode="opEditGridView" forGrid="grid"/>
```

7.7.4 opNewGridView

Die Operation NewGridView öffnet ein Dialogfeld zur Erstellung einer neuen Ansicht auf Basis der aktuellen Ansicht des zugeordneten Grid-Elements. Wenn forGrid weggelassen wird, dann wird das fokusierte Grid genommen, d.h. es wird auch die Tool-Id dieses Grid übernommen.

Die Operation kann in Menüs, PopupMenüs und ToolbarButtons verwendet werden.

Beispiel:

```
<Operation ID="ViewNew" opCode="opNewGridView" forGrid="grid"/>
```

7.7.5 opDeleteGridView

Die Operation DeleteGridView löscht (nach Benutzerrückfrage) die aktuelle Ansicht des zugeordneten Grid-Elements.

Die Operation kann in Menüs, PopupMenüs und ToolbarButtons verwendet werden.

Beispiel:

```
<Operation ID="ViewDelete" opCode="opDeleteGridView" forGrid="grid"/>
```

7.7.6 opViewsReload (erst ab Byron/BIS v4.8.0)

Die Operation ViewsReload lädt die Ansichten eines Grids neu, dh. die FilterNavigation des Elements GridView wird erneut ausgeführt. Ferner kann über diese Operation eine Ansicht vorgewählt werden.

Beispiel:

```
<Operation ID="reloadViews" opCode="opViewsReload" forElement="grid1"
           default="TheNameOfViewDesired">
</Operation>
```

8 Gantt

Mit dem Applikations-Element "Gantt" kann eine Ganttdarstellung von Objekten konfiguriert werden. (Hinweis: Das Gantt-Element beinhaltet auch ein Grid-Element)

8.1 Element: GanttView

Definiert eine oder mehrere Gantt-Ansichten, welche verwendet werden sollen.

8.1.1 Parameter für die Filternavigation (>= v4.11.6)

Die Filter-Navigationen in den Gantt-Ansichten unterstützen folgende Parameter

Parameter	Datentyp	Bemerkung
VAL_FirstDate	Datum	Erstes Datum im Kalender
VAL_LastDate	Datum	Letztes Datum im Kalender

8.2 StatusMapEntry: GanttHint

Zeigt den Hint in der Statusbar an, wenn sich die Maus über ein Gantt-Objekt befindet.

Falls eine Gantt-Beschriftung im `DescriptionName`-Element angegeben wurde, wird diese berücksichtigt.

Siehe StatusMapping

Beispiel

```
<StatusMapping>
    <StatusMapEntry ID="GanttHint" panel="3"></StatusMapEntry>
</StatusMapping>
```

8.3 Gantt-Operationen

8.3.1 Operation: opGoTo

Gehe zu Datum

Beispiel:

```
<Operation ID="GoTo" opCode="opGoTo" caption="Gehe zu Datum..." forElement="gantt"/>
```

8.3.2 Operation: opGoToNow

Gehe zu Heute

Beispiel:

```
<Operation ID="GoToNow" opCode="opGoToNow" caption="Gehe zu Heute" forElement="gantt"/>
```

8.4 Beispiel

```
<Gantt ID="gantt" synchGridSelection="true" panel="pnGantt" onDoubleClick="Open"
       containerSupport="true">
    <Contents>
        VIA bisR_ReservableObjectContainerToContents CLASS Room
    </Contents>
    <GridView userinterface="false" includeStandard="true"></GridView>
    <GanttView>
        START INSTANCES bisB_GanttView VALUE bisB_ToolId = "ApReservGanttExpl"
    </GanttView>
    <StatusMapping>
        <StatusMapEntry ID="GridContents" panel="1"/>
        <StatusMapEntry ID="GridAutoSum" panel="2"/>
        <StatusMapEntry ID="GanttHint" panel="3"/>
    </StatusMapping>
    <!-- Popup nur für Gantt -->
    <Popupmenu></Popupmenu>
</Gantt>
```

9 Graphic

Mit dem Application Element <Graphic> können Objekte grafisch dargestellt werden. Die Darstellung erfolgt aufgrund der Attribute:

- Symbol an einer Position.
  zwingende Attribute: Position, bisG_SymbolName (und Symbol muss existieren).
  weitere Attribute: Z_Drehwinkel, bisG_Scale, bisG_YStretch
- Flächen. zwingende Attribute: bisB_AreaGeometrie
- Verbindungslinien. (erst ab Byron/BIS v4.11.3)
  zwingende Attribute: bisB_Jointer
  weitere Attribute: bisB_JointType, bisG_Width, bis_Color.

Spezielle Attribute von Graphic:

```
<Graphic panel="pnGraphic"
         priorityAttribute="MV_PickPriority"
         condorDisplayAttribute="$Object_Type$:   $Object_Display_Kontext$"
         dropTopOnly = "true"
         zoom = "true"
         processGraphicChangeToDb ="true"
         pickBox="6" />
```

`priorityAttribute` und `dropTopOnly` vgl Element Layer.

`condorDisplayAttribute` steuert die Darstellung des Objektes im ‚on over hint' in der ToolBar status (`<ShowToolBar status="true"/>` und bei Selection in der Statusbar, falls entsprechend konfiguriert (vgl. StatusMapping 'Selection').

Mit `processGraphicChangeToDb` steuert man, ob eine Grafikänderung analysiert und in die Datenbank geschrieben werden soll. (default="true") Durch diese Technik können Standard Condor Operationen genutzt werden. Durch Einblenden der Toolbars:

```
<ShowToolBar action="true" />
```

Oder durch entsprechende Operationen, zB. grOBJECT_MOVE etc.

Mit `zoom = "false"` wird der eingestellte Ausschnitt beibehalten, wenn neue Objekte angezeigt werden.

Mit `processGraphicChangeToDb="false"` können die Daten gegen Änderungen geschützt werden.

pickBox definiert die Initalgrösse des Pick-Quadrates. Vgl. custom propagate `"PickBoxSize"`

9.1 Implementierung StatusMapping

Das Application Element Graphic implementiert eine Ausgabe auf den Statusbar (vgl. StatusMapping). Durch ein StatusMapEntry mit der ID `Selection` wird der Text des selektierten Elements in der Statusbar angezeigt.

Bei der Selektion wird `condorDisplayAttribute` auf ‚Selection' angezeigt. Bei Aktionen zB. ‚verschieben' werden die erwarteten Befehle ebenfalls auf ‚Selection' angezeigt.

Beispiel

```
<StatusMapping>
    <StatusMapEntry ID="Selection" panel="1"></StatusMapEntry>
</StatusMapping>
```

9.2 Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
OBJ_CurrentView	Objekt	Aktuelle Ansicht (nur wenn auch ein Objekt verwendet wird)
VAL_CurrentView	Text	Titel der aktuellen Ansicht. Funktioniert auch für konfigurierte Ansichten.

Ab Version 5.1.2

9.3 Element: Background

Enthält eine Filternavigation zu den Hintergrundsplänen (Doc_File bzw. Ableitung davon). Als Eingangsobjekt wird das zu ladende Objekt verwendet (zB. Das Stockwerk).

Beispiel

```
<Background>
    CLASS Floor VIA Doc_Documents CLASS Doc_File
</Background>
```

Der Hintergrund kann auch in GraphicView(s) definiert werden.

9.4 Element: Condition

Condition enthält die Bedingung in Form einer Filternavigation, ob ein load durchgeführt werden soll. Wenn das Eingangsobjekt die Bedingung nicht erfüllt (Ausgangsmenge leer), dann wird das Bestehende beibehalten.

Beispiel: wenn ein Stockwerk als Eingangsobjekt auftritt, dann wird dieses dargestellt. Bei allen übrigen Objekten geschieht nichts.

```
<Condition>CLASS Floor</Condition>
```

9.5 Element: Contents

Bestimmt die darzustellenden Objekte.

Beispiel

```
<Contents>
    CLASS Floor VIA Object_To_Children CLASS Room
    [ OR VIA Room_To_Component]
</Contents>
```

9.6 Element PaperColor

Bestimmt die Farben des Papiers.

```
<PaperColor inside="#FFFFFF" outside="#E0E0FF" margin="#E5E5E5"/>
```

9.7 Element Pick

Pickbarkeit der Fächen (shapes) und Positionskomponenten (symbols). Mit solid wird definiert, ob die Flächen beim Markieren rot werden, oder ob nur die Umrandung gefärbt wird.Note: s

```
<Pick shapes="false" symbols="true" solid="false"/>
```

Note: symbols / solids sind obsolet da dies mit dem Element Layer definiert werden können.

```
<Pick shapes="false"/>
```

```
<Layer shapes="false"/>
```

9.8 Beschriftungsposition (ab Byron/BIS V5.3.2)

Die Position und der Stil der Beschriftung ist im binären Attribut `bisG_Beschriftung` gespeichert. Das Attribut kann als Text gelesen und geschrieben werden. Man kann das Attribut durch Filternnavigation verändern:

```
MODIFY bisG_Beschriftung 'TRP:LO'
```

Setzt den Textreferenzpunkt links-oben.

Um diese Werte bequemer aus Listen zu manipulieren kann ein Text Klassenn-Attribut mit einer Getter und Setter definiert werden.

bisG_BeschriftungAsTxt (Beschriftung als Text) Kllassennattribut; Lesen&schreiben

Lese-Methode:

```
method return Text is
  res: Text;
  x : boolean;
begin
  Get_Text_Value(Search_View("bisG_Beschriftung"), self, "", x, res);
  if x then
    return res;
  end if;
end;
```

Schreibe-Methode:

```
method (inout action: Integer; inout flags: Integer; inout value: Text) is
  i : integer;
begin
  if IsBitSet(flags, 0) then
    i := Modify_Text_Value(Search_View("bisG_Beschriftung"), self, value, "");
  end if;
  action := 2;
end;
```

Der Text besteht einer Liste von <Option> : <Wert> ;

Beim Schreiben müssen nicht alle Optionen spezifiziert werden. Nicht definierte Optionen bleiben unverändert. Nach dem letzten Wert kann der ';' weggelassen werden. Bei Koordinaten muss immer X und Y Wert angegeben werden.

Option	Beschreibung	Mögliche Werte
TRP	Text Rereferenzpunkt der Linie	Positionscodierung 1)
ORP	Objekt Rereferenzpunkt der Linie	Positionscodierung 1)
L	Referenzlinie	J oder N für mit bzw. ohne Referenzlinie
S	Sichtbare Beschriftung	J oder N für sichtbare bzw ausgeblendete Beschriftung
TDX	Offset von Objekt Oben-Links zu Text Unten Links	X-Wert in Meter
TDY	wie TDX	Y-Wert in Meter
ODX	Zusätzlicher Offset des Linienendpunktes auf der Objektseite 2)	X-Wert in Meter
ODY	wie ODX 2)	Y-Wert in Meter
ROT	Drehwinkel der Beschriftung (bisG_Null-Rot)	Float als Radiant

1) Positionscodierung besteht aus zwei Zeichen, zuerst die horizontale Position L, M oder R für Links Mitte oder Rechts. Das zweite Zeichen definiert die vertikale Position O, M, oder U für Oben, Mitte oder Unten.
2) Bis zur Version 5.3.2 gab es nur ORP:OL. Die Referenzlinie wurde dann mit ODX/Y auf der Objektseite angepasst. Bis dahin kann also Objektreferenzpunkt Mitte so aussehen:
ORP:LO;ODX:1.300000;ODY:-3.105000;
wenn dann der Objektreferenzpunkt erneut in die Mitte gesetzt wird, sieht dies so aus:
ORP:MM;ODX:0.000000;ODY:0.000000;
Das Zurücksetzen von ODX/ODY auf 0.0 erfolgt nur beim Umsetzen vom alten ins neue Format.

Beispiel:
`TRP:MU;ORP:MM;L:J;S:J;TDX:-0.003379;TDY:-0.540959;ODX:-0.200000;ODY:1.000000;ROT:0.380000000`

9.8.1 Beschriftungsinformation als Einzelattribute (ab Byron/BIS v5.6.1)

Die Bestandteile der Beschriftung können nicht nur – wie oben beschrieben – durch einen Text angezeigt und manipuliert werden, sondern auch durch einzelne Attribute. Dafür wurden folgende Pseudoattribute für die Klassen *Space* und *PositionComponent* eingeführt.

Identifier	Bezeichnung	Bemerkung
bisG_BeschriftungDx	Beschriftung: Offset X	Float
bisG_BeschriftungDy	Beschriftung: Offset Y	Float
bisG_BeschriftungHidden	Beschriftung: versteckt	Boolean
bisG_BeschriftungLine	Beschriftung: zeige Linie	Boolean
bisG_BeschriftungObjRef	Beschriftung: Linienansatz beim Objekt	Siehe auch vorheriges Kapitel Links, Oben = 0 Links, Unten = 1 Links, Mitte = 2

		Rechts, Oben = 3 Rechts, Unten = 4 Rechts, Mitte = 5 Mitte, Oben = 6 Mitte, Unten = 7 Zentrum = 8
bisG_BeschriftungRot	Beschriftung: Drehung	Radiant (pi/2 = 90°)
bisG_BeschriftungTextRef	Beschriftung: Linienansatz beim Text	Siehe bisG_BeschriftungObjRef. Der Wert 8 kann hier nicht verwendet werden.

9.9 Element EditText (ab Byron/BIS v4.11.0)

Steuert die Berechtigung für das Verändern der Textbeschriftungen. Wenn man diese Berechtigung hat, dann erhält man Textbeschriftungen verschieben und erhält ein Popup-Menu, wenn Beschriftungen gewählt sind. Das Menu erlaubt das ein und ausblenden der Referenzlinien, Anschlusspunkte der Referenzlinie auf Seite des Objektes und der Beschriftung, das Wählen des Objektes und das Ausblenden der gesamten Beschriftung. Die letzten zwei Befehle haben eine ‚Gegenoperation' im ‚normalen' Popup-Menu: *‚Beschriftung wählen'*. Wenn es die Beschriftung gibt, dann wird diese selektiert, sonst wieder eingeblendet.

Ohne Angabe gilt der Default:

```
<EditText groupRight="true"/>
```

Damit wird die Berechtigung durch das Gruppenrecht *‚Grafik: Beschriftung ändern'* gesteuert.

```
<EditText groupRight="false"/>
```

Damit gibt es für alle keine Berechtigung.

```
<EditText groupRight="false">
    START USER VALUE bisB_UserFullname 'Ei*'
</EditText>
```

Damit wird die Berechtigung nur durch die Filternavigation gesteuert. Die Bedingung ist erfüllt, wenn mindestens ein Objekt resultiert. Die Navigation wird ohne Eingangsobjekt aufgerufen.

```
<EditText groupRight="true">
    START USER VALUE bisB_UserFullname 'Ei*'
</EditText>
```

Damit sein alle berechtigt, die das Gruppenrecht besitzen **oder** die Navigation erfüllen.

Note: Beschriftungstext pickbar wird dadurch ermöglicht:

```
<ToolbarButton><Execute name="G_TxtPick" /></ToolbarButton>

<Operation ID="G_TxtPick" opCode="grPickable" forGraphic="Graphic"
           priorityLayer="800"
           caption="@grTextPickable_Caption@"
           icon="grDescriptionPickable"/>
```

9.10 Element Layer

Damit werden Layer definiert. Jedes Objekt liegt auf genau einem Layer. Diese haben zwei Funktionen:
1. Die Pickbarkeit wird über den Layer gesteuert. (vgl. Operation `grPickable`)

2. Der Z-level der Objekte definiert sich über die priority des Layers auf welchem das Objekt liegt. Objekte mit höherer Priorität liegen über den Objekten mit niederer priority. Neben der Sichtbarkeit hat dies einen Einfluss bei der Bestimmung des Drop-Targets. Mit `dropTopOnly` steuert man ob man nur das Objekt mit der höchsten Priorität (true) oder alle (false) haben möchte. Im letzten Fall kann man in der Target Filternavigation auch `SORT (-graphicPriority) PICK 0 1` einbauen.

Die Priorität der Objekte bestimmt sich aus dem Integer-Attribut, welches in priorityAttribute am Element Graphic angegeben ist. Bsp, wenn demo_graphicPriority ein integerAttribut ist:

```
<Graphic priorityAttribute="demo_graphicPriority" dropTopOnly="true" />
```

Wenn kein Wert am Objekt definiert ist, gilt der Default:

Für Flächenobjekte:	200
Für Verbindungsobjekte:	300
Für Positionsobjekte:	400
Beschriftungen:	800

Pickbarkeit der Fächen (shapes) und Positionskomponenten (symbols). Mit solid wird definiert, ob die Flächen beim Markieren rot werden, oder ob nur die Umrandung gefärbt wird.

```
<Layer name="BisLayerFloor" priority="200" />

<Layer name="BisLayerRoom" priority="201" />

<Layer name="BisLayerZone" priority="202" />
```

In diesem Beispiel müsste an der Klasse Room das Attribut `demo_graphicPriority` definiert sein, mit einem Vorgabewert von 201.

Das Attribut `name` kann weggelassen werden. Der Name des Layers hat in der Konfiguration keine Bedeutung und ist höchstens bei einem Export ersichtlich.

Wenn der Benutzer mehrere Objekte pickt, dann werden nur die Objekte mit der grössten Priorität gewählt. zB. wenn eine Komponente in einem Raum liegt und der Benutzer waählt die Komponente und damit auch den darunter liegenden Raum, dann wird der Raum nicht selektiert, da er einen niedrigere Priorität hat. Dieses Verhalten kann man mit der Alt-Taste aufheben. So kann man zB. alle Objekte in einem Rechteck wählen, indem man eine Ecke wählt, das Rechteck aufzieht und vor dem Loslassen der Maustaste noch die Alt-Taste drückt. Das funktioniert auch in Kombination mit additiver Selektion (Ctrl-Taste)

9.11 Element ShowToolBar

Definiert, welche Condor Toolbars eingeblendet werden.

```
<ShowToolBar status="true" zoom="false" file="false" symlib="false" action="false"
             attribute="false" draw="false" control="false"/>
```

9.12 Element Color

Definiert, die Farben der Flächen. Diese kann auch in GraphicView(s) definiert werden.

Verwendung einer Färbung:

```
<Color>
    START INSTANCES BisG_FAinstance VALUE Name = Nutzung
</Color>
```

oder den RGB Wert der Farbe:

```
<Color fix="#0000E0"/>
```

9.13 Element PlanToObject / ObjectToPlan

Diese zwei Elemente sind nur von Bedeutung, wenn an den Gebäuden / Stockwerken ein Koordinatensystem definiert ist (bisG_NullX, bisG_NullY, bisG_NullRot). Das Koordinatensystem wird vom Layout-Editor wie folgt interpretiert: Die Objekte (Räume, Komponenten) werden um diese Transformation ‚zurück' verschoben/gedreht und passen dann zum ‚lokalen' Architekturplan. Die Objekte sind mit globalen Koordinaten abgelegt.

Dieses Verhalten erreicht man durch Definition des gewünschten Koordinatensystems:

```
<ObjectToPlan>
    RECALL OBJ_StartObjects CLASS Space
</ObjectToPlan>
```

Wenn nichts definiert wird, dann wird das StartObjekt genommen, sofern es genau eines gibt. Damit erübrigt sich obige Definition im Allgemeinen. Um dieses default Verhalten zu unterdrücken kann zB. `<ObjectToPlan>BLOCK</ObjectToPlan>` geschrieben werden.

Möchte man Objekte aus verschiedenen Koordinatensystemen gleichzeitig anzeigen, dann muss obiger Mechanismus ausgeschaltet werden. Dies ist Explizit notwendig, wenn es genau ein Startobjekt gibt und dieses ein ‚falsches' Koordinatensystem besitzt. Möchte man nun dieselben (Stockwerk) Pläne verwenden, dann müssen dies zu den Objekten hin verschoeb werden. Das System muss also wessen zu welchem Koordinatensystem (Stockwerk / Gebäude) der Plan gehört. Dies kann wie folgt definiert werden:

```
<PlanToObject>
    VIA Doc_Referenced_By CLASS Space
</PlanToObject>
```

Dies entspricht der Standard-Modellierung in Byron/BIS

9.14 Element ComponentLegend

Definiert das Aussehen der Komponetenlegende. Deie Default Werte sind:

<ComponentLegend height="0.8" width="2.5" vertical="true" maximum="2" title="Legende" visible="true" />

Die Komonentenlegende wird nur dargestellt, wenn < ComponentLegend/> definiert ist, und wenn im Rahmenplan ein Rechteck für die Legende definiert ist. Vgl Element GraphicViews - Layout

9.15 Element GraphicViews

Definiert die Ansicht (dargestellte Objekte, Farbe, Hintergrundsplan und Beschriftung).

Bestehende Grafikansichten können mit <Use> verwendet werden. Mit <GraphicView> kann auch eine Ansicht ad-hoc definiert werden. (siehe auch Grafik-Ansichten)

Ansichten können durch den Benutzer umgestellt werden (userinetrface=true) oder durch ein Custom-Propagate. Die `caption` wird in der Combo-Box angezeigt, der `name` wird vom Custom-Propagate verwendet.

NavContents definiert die darzustellenden Objekte. Das Eingangsobjekt wird durch inputIs definiert:

nothing:	«<nichts>»: die Navigation wird gar nicht ausgeführt, es kommen keine zusätzlichen Objekte durch die Navigation.

showedObjects: «dargestellte Objekte»: alle Objekte, welche durch Start -> Contents gezeichnet werden konnten.

navigatedObjects: «erreichte Objekte»: alle Start -> Contents Objekte

startObject: «Ausgangsobjekt»: nur das Startobjekt.

Plan definiert die Hintergrundspläne. Ausgangsobjekt der Filternavigation ist das Startobjekt (z.B. Floor). Wenn das erreichte Objekt ein Tool ist, dann muss es ein Layout-Editor sein und der Hintergrund wird aus dem Dokument gelesen (Layout-Editor – Hintergrund). Wenn das Objekt ein Doc_File ist, dann wird der Hintergrund aus der Datei gelesen. Wenn für minSizeForThread ein Wert grösser als 0 angegeben wird, dann erfolgt das Lesen in einem separaten Thread (im Hintergrund parallel zur sonstigen Verarbeitung).

Label definiert die Grafik-Beschriftung (bisG_TextContainer).

Color definiert die Färbung (bisG_FAInstance). Mit dem Attribut fix kann direkt eine Farbe definiert werden, dies anstelle einer Filternavigation zu einer Färbung.
<Color fix="#1010E0"/>

Layout definiert den Rahmenplan. Das Plandokument wird mit einer Filternavigation angegeben. Im Plan werden Rechtecke definiert, mit erweiterten Daten 'BisRahmen' Wert 'P' bzw. 'K':

Block 'BisRahmen' Text 'P' definiert den Ort für den Plan,
Block 'BisRahmen' Text 'K' definiert den Ort für die Komponentenlegende,
Das Aussehen der Komponentenlegende wird in Element ComponentLegend definiert.

Hatch definiert den Stil der Raumgeometriedarstellung. Style = solid, line oder hollow. (Default ist: `<Hatch style="solid"/>`)

solid: die Polygone werden gefüllt dargestellt.

hollow: es wird nur die Umrandung gezeichnet, auch bei der Selektion.

line: es wird schraffiert gezeichnet. In diesem Fall beinhaltet Hatch noch ein Element Line.

Line definiert den Linienstil im Fall `style="line" mit den Attributen width= Dicke, widthMode=0` Dicke in Pixel, `widthMode=1` Dicke in mm Papier, `widthMode=2` Dicke in Meter (Welt). `angle=` Winkel der Schraffur in Grad

Beispiel, welches auch den default Werten entspricht:

```
<Hatch style="line">
  <Line width="1" widthMode="0" distance="0.15" angle="45" />
</Hatch>
```

```
<GraphicViews comboWidth="290" userinterface="true">
    <Use>START INSTANCES bisG_GraphicView</Use>
    <GraphicView caption="Standard Info" name="std">
        <NavContents inputIs="showedObjects">
            VIA "Room_To_Component"
        </NavContents>
        <Plan minSizeForThread="20">
            VIA Doc_Documents
            CLASS Doc_File
            VALUE Doc_Path "*GRUNDRISS.drw"
        </Plan>
        <Label>
            START INSTANCES bisG_TextContainer VALUE Name 'Rauminformation'
        </Label>
        <Color>
            START INSTANCES bisG_FAInstance VALUE Name 'DIN*'
        </Color>
    </GraphicView>
</GraphicViews>
```

Ansichten können mit `<SubGraphicView>` in Untermenüs gruppiert werden ab 4.9.11

```
<GraphicViews comboWidth="290" userinterface="true">
    <GraphicView caption="Standard Info" name="std"> … </GraphicView>
    <SubGraphicView caption="Standard Info">
        <Use>START INSTANCES bisG_GraphicView VALUE Name "Raum*"</Use>
        <GraphicView caption="Standard Info" name="std"> … </GraphicView>
    </SubGraphicView>
</GraphicViews>
```

9.15.1 Grafik-Ansichten

Die mit <use> verwendbaren Grafikansichten werden im Werkzeug "Konfiguration" unter "Grafik-Ansichten" konfiguriert.

Die Objekte, auf welche im Reiter "Dargestellte Objekte" unter "Ausgehend von" Bezug genommen wird, ist analog zu denen unter NavContens.

<nichts>	(nothing) Die Navigation wird gar nicht ausgeführt, es kommen keine zusätzlichen Objekte durch die Navigation.
dargestellte Objekte	(showedObjects) alle Objekte, welche durch Start -> Contents gezeichnet werden konnten. Kommen keine darstellbaren Objekte gemäss der Filternavigation unter <Contents> in bei der Grafik-Ansicht an (z.B. das Geschoss), so wird die Navigation für "Dargestellte Objekte" nicht ausgeführt.
erreichte Objekte	navigatedObjects: alle Start -> Contents Objekte
Ausgangsobjekt	startObject: nur das Startobjekt. Die Navigation unter <Contents> wird nicht aufgeführt.

9.16 Grafik Operationen

Das Drucken einer Grafik mit der Operation grPrinter kann auch ohne Grafikelement erfolgen und ist deshalb im Kapitel Operation beschrieben.

Die Grafik-Operationen unterstützen ein forGraphic, damit kann der Befehl unabhängig vom Fokus.

Beispiel:

```
<Operation ID="grAreaPick" opCode="grAreaPickable" forGraphic="Graphic"/>
```

S: Schalterbefehl, d.h. der Befehlt schaltet die Einstellung um (angehakt <-> nicht angehakt)

A: Befehl der weitere Eingaben erfordert. Diese Befehle können mit dem Befehl grCancel abgebrochen (A) werden.

D&D Drag/Drop Befehl

9.16.1 Operation: grCancel

Ein A Befehl wird abgebrochen.

9.16.2 Operation S: grAreaPickable

Shapes, das sind Flächendarstellungen von Räumen sind pickbar, bzw. nicht pickbar. Die Operation ist ein Umschalter (checked Haken).
obsolet vgl. grPickable.

```
<Operation ID="x" opCode="grPickable" forGraphic="Graphic" priorityLayer="200"
           icon="Space"/>
```

9.16.3 Operation S: grSymbolPickable

Blöcke, das sind Symboldarstellungen von Komponenten sind pickbar, bzw. nicht pickbar. Die Operation ist ein Umschalter (checked Haken).
obsolet vgl. grPickable

```
<Operation ID="x" opCode="grPickable" forGraphic="Graphic" priorityLayer="400"
           icon="Component"/>
```

9.16.4 Operation S: grPickable

Objekte von einem bestimmetn Layer pickbar schalten vgl. Element Layer. Der Layer wird durch seine priority identifiziert (`priorityLayer=`). Mehrere Layer können durch ; getrennt angegeben werden oder ganze Bereiche mit -
Bsp "150;152;160-165;170;180-199"
Der Initialwert kann durch pickable=<boolean> gesetzt werden (default = "true").

```
<Operation ID="x" opCode="grPickable" forGraphic="Graphic" priorityLayer="195-205"
           icon="Space"/>
```

9.16.5 Operation S: grSolidHighlight

Selektierte Flächen werden entweder durch das Einfärben der Fläche oder nur der Umrandung hervorgehoben.

initialDown: Steuert den Anfangszustand des Schalters (gedrückt bzw. ungedrückt).
Default: "false" ab V 5.1.0

9.16.6 Operation S: grLineWidth (erst ab Byron/BIS v4.11.3)

Die Darstellung der Verbindungen kann umgestellt werden. Die Operation ist ein Schalter, wenn er gedrückt ist, dann wird die eingestellte Farbe und Liniendicke (in Pixel) für alle Verbindungsobjekte verwendet. Ohne den Schalter wird die Liniendicke vom Attribut bisG_Width (in Meter) gelesen und die Farbe vom Attribut bis_Color, d.h. von jedem Objekt individuell.

Beispiel der Operation:

```
<Operation ID="x" opCode="grLineWidth" forGraphic="Graphic"
           width="5" initialDown="true" color="#000000" icon="grOrthoOff"/>
```

Bedeutung der Attribute

width: Dicke der Verbindung in Pixel.
Default: Attribut bisG_Width (in Meter) am Objekt.

color: Farbe der Verbindung als RGB HEX-code.
Default: Attribut bis_color am Objekt.

initialDown: Steuert den Anfangszustand des Schalters (gedrückt bzw. ungedrückt).
Default: "false"

9.16.7 Operation: grZoomPage

Der Ausschnitt wird auf das Blatt gesetzt.

9.16.8 Operation: grZoomTotal

Der Ausschnitt wird so gesetzt, dass alle Objekte sichtbar sind.

9.16.9 Operation A: grZoomDetail

Der Zoom-Befehl wird initiiert. Es wird ein Fadenkreuz angezeigt, damit der Benutzer den ersten Eckpunkt eingeben kann. Danach folgt ein Rechteck um den Ausschnitt auszuwählen.

9.16.10 Operation D&D: grDropPosition

Setzt das Attribut Position auf die Koordinate des Drop Punktes. Wenn die Operation noch ein <Bind> enthält, dann wird noch eine Assoziation auf die Fläche hergestellt. <Source>, <Target> und ‚forGraphic' werden ebenfalls unterstützt.

ab V 4.9.12: Es können auch mehrer Objekte gleichzeitig verschoben werden. Dieses verhalten kann auch durch den mode gesteuert werden `mode = "positionOnly", "moveOnly", "both"`
Default `"both"`: Damit wird mit einem Objekt positioniert und bei mehreren Objekten verschoben.

ab V 5.0.0: Ab dieser Version wird kein Pfeil mehr gezeichnet, ausser es wird durch das Attribut `echo="arrow"` verlangt.

Beispiel:

```
<Operation ID="G_DropPosition" opCode="grDropPosition"
           caption="grDropPosition_Caption" forGraphic="Graphic">
    <Bind assoc="Room_Of_Component"/>
</Operation>
```

9.16.11 Operation: grEditArea

Startet den Flächen Editier Dialog. Entgegen dem normalen verhalten definiert das Attribut `forGraphic` nur den Ort, wo das Editieren stattfindet. Das Objekt (Source) wird durch das Attribut forElement definiert, im folgenden Beispiel fehlt es, damit kann der Raum in jedem Appl.Element gewählt werden.
Es gibt eine Default Caption `@grEditArea_Caption@`

```
<Operation ID="G_EditArea" opCode="grEditArea" forGraphic="Graphic">
    <Source>CLASS Room</Source>
</Operation>
```

9.16.12 Operation: grMeasure ab V 5.2.1

Startet eine Messaktion. Mit dem ersten Mausklick wird der Startpunkt bestimmt. Danach beginnt die Messung. Mit der Shift-Taste kann der Orthomodus aktiviert werden. Der zweite Mausclick beendet die Messung. Mit Backspace wird die Aktion um einen Schritt zurück versetzt, d.h. es kann entweder ein neuer Startpunkt gewählt werden, oder die Aktion wird abgebrochen.

Während der Messung wird das Resultat auf dem Statuspanel ausgegeben, dieses muss also konfiguriert sein, sonst sind diese Werte nicht sichtbar:

```
<StatusMapping>
  <StatusMapEntry ID="Selection" panel="1"></StatusMapEntry>
</StatusMapping>
```

Beispiel:

```
<Operation ID="G_Measure" opCode="grMeasure" forGraphic="graphic"
  moveFormat="Distanz %0:.2f m"
  resultFormat="Distanz %0:.2f m   (%1:.2f  %2:.2f) "
  showMessage="true"
```

```
  leaveEcho="false"
/>
```

`moveFormat`
Definiert den Text während des Messens. Es stehen folgende Werte zur Verfügung:
%0: Distanz
%1: Dx gemäss Ansicht
%2: Dy gemäss Ansicht
%3: Dx gemäss Weltkoordinaten
%4: Dy gemäss Weltkoordinaten
Default: `moveFormat="%0:.2f m  (%1:.2f  %2:.2f"`

`resultFormat`
Definiert den Text der am Schluss des Messens angegeben wird.
Default: gleicher Wert wie `moveFormat`

`showMessage boolean`
Steuert ob am Schluss noch eine MessageBox (vgl. resultFormat) gezeigt wird
Default: true

`leaveEcho boolean`
Steuert ob am Schluss der Echo Pfeil sofort entfernt oder stehen gelassen wird und erst beim nächsten Neuzeichnen verschwindet.
Default: true

9.16.13 Operation: grMoveCopyArea

Startet den "Flächen kopieren/verschieben"-Dialog. Entgegen dem normalen Verhalten, definiert das Attribut `forGraphic` nur den Ort, wo das Editieren stattfindet. Die Objekte (`Source` und `MandatorySource`), welche verschoben/kopiert werden sollen, werden durch das Attribut `forElement` definiert.

Diese Operation verhält sich analog zu `opCopyTemplate`. `Source` enthält die Objekte, welche im Dialog entfernbar sind und `MandatorySource` enthält von `Source` abhängige Objekte welche auch mitkopiert werden müssen. Das Element `MandatoryVisible` kann dazu verwendet werden, einige mitkopierte Objekte nicht im Grid darzustellen. Es werden nur die `MandatoryObjekte` dargestellt, welche durch die `MandatoryVisible`-FilterNavigation nicht rausgefiltert werden.

Die Inputs sind wie folgt: `Source` -> `MandatorySource` -> `MandatoryVisible`

Folgende Attribute sind zulässig:

`forGraphic`, `moveOnly` (default=false), `deepCopy` (default=true), `orthoMode` (default=false, kann leer bleiben dann wird aus Registry gelesen), `allowChangeOrthoMode` (default=true), `allowChangeMoveOnly` (default=true), `allowChangeRenameFormula` (default=false), `offsetX` (default=0), `offsetY` (default=0), wenn Offsets angegeben geht Operation nicht automatisch in "Vektorselektier-Modus".

ab V 5.0.0: Ab dieser Version wird kein Pfeil mehr gezeichnet, ausser es wird durch das Attribut `echo="arrow"` verlangt.

9.16.14 Operation: grPrint

ab V 4.8.6 Grafik Drucken. Die Blattränder können mit folgenden Attributen definiert werden (Angaben in mm) mrgLeft, mrgRight mrgTop. Und mrgBottom. Mit mrgHeader und mrgFooter wird der Abstand der Kopf-Fusszeile vom inneren Rand definiert 1 = 10% der Zeichenhöhe. Das Attribut style kann die Werte condor, part oder full haben. Zwischen Ausschnitt drucken (part) und ganze Seite

(full) kann der Benutzer noch im Formular umstellen. Condor Drucken ergibt das Ausschnittdrucken von Condor.

Mit Text können Beschriftungen platziert werden vertical=header, oder footer, horizontal = left, right oder center. Dabei wird <Mst>durch den Massstab <Datum:format> durch das Datum und <REM> duch eine interaktiv eingegebene Bemerkung ersetzt. Vgl. nachfolgende Beispiele.

Mit `<Legend>` wird die Farblegende am unteren rechten Rand des Blattes platziert. ab 4.9.6 ab 4.10.3 sind noch folgende EInstellunegn möglich: `<Legend fontName="Verdana" fontHeight="20" lineSpacing="30" colorSpacing="50" mrgRight = "10" />` Der Zeilenabstand (lineSpacing) wird prozentual zur Texthöhe angegeben (zB. 50: halbe Texthöhe Abstand zwischen den Zeilen) colorSpacing definiert ebenfalls prozentual den Abstand zwischen dem Farbquadrat und dem Text. Mit mrgRight wird der Abstand der Legende vom rechten Rand definiert. Alternativ kann man auch mrgLeft angeben. Die Angaben jeweils in Millimeter.

Mit Var können Variablen definiert werden, welche dann in den Texten eingesetzt werden können, wie REM und Datum. Das Ausgangsobjekt der Navigation ist das Startobjekt. ab 4.9.9

Der Befehl hat folgende default Werte, ausser <Text>

Beispiel:

```
<Operation ID="GPrint" opCode="grPrint"
           style="part"
           mrgLeft= "11" mrgRight= "11"
           mrgTop= "11" mrgBottom= "11"
           mrgHeader= "2" mrgFooter= "2">
    <Font name="Tahoma" height="40"/>
    <Var name="Var" attribute="Object_Display_Kontext"/>
    <Var name="g" >VIA Object_To_Parent</Var>
    <Text vertical="header" horizontal="left">&lt;g&gt;  &lt;Var&gt;</Text>
    <Text vertical="header" horizontal="right">&lt;REM&gt;</Text>
    <Text vertical="footer" horizontal="left">Massstab &lt;Mst&gt;</Text>
    <Text vertical="footer" horizontal= "right">gedruckt &lt;Datum:d.m.yyyy&gt;</Text>
    <Legend fontName="Verdana" fontHeight="20"
            lineSpacing="30" colorSpacing="50" mrgLeft = "2" />
</Operation>
```

9.16.15 Operation: grMove

ab 4.9. <Source> kann als Bedingung verwendet werden.
Damit können mehrere Objekte Räume / Komponenten) verschoben werden. Es wird ein Start- und ein Endpunkt verlangt, daraus ergibt sich die Verschiebung. Mit der Shift Taste schaltet sich der Ortho-Modus ein. Um Objekte mit verschiednen Pick-Prio zu verschieben muss die ‚Alt' Teste verwendet werden. zB. Box-Pick mit der ‚Alt' Taste bei der 2. Ecke kann ein Raum mit samt dem Rauminhalt kopiert werden. Dieser Befehl hat eine funktionale Überlappung mit grMoveCopyArea. Der Befehl führt die Änderungen auf der Grafik aus. Die Änderung in der Daten bank erfolgt über das 'Geometrie-hat-sich-verändert-Ereignis'. Der Befehl funktioniert also gleich wie die Standard Condor Befehle, zB. grOBJECT_MOVE. Durch processGraphicChangeToDb ="false" funktioniert der Befehl nicht mehr.

Das Attribut snap = off, background, full steuert ob es einen Fangmodus gibt und ob dieser nur auf dem Hintergrund oder auf Hintergrund und Vordergrund (Bis-Objekte) funktioniert. Letzteres kann bei bestimmten Symbolen zu erheblichen Verzögerungen führen. Der Default ist background:

```
<Operation ID="GMove" opCode="grMove" snap="background">
  <Source>CLASS = "Printer"</Source>
</Operation>
```

ab 4.10.6 Die Operation kann auch ein Bind Objekt enthalten:

```
<Bind assoc="Room_Of_Component"/>
```

In diesem Fall wird an der Zielposition ein Flächenobjekt gesucht. Dieses wird dann als Partner für die Assoziation verwendet. Wenn eine Target Navigation deklariert ist, wird das Resultat dieser verwendet. Damit ist es möglich eine Komponente mit dem Stockwerk zu verknüpfen, falls diese ins ‚Leere' gezogen wird.

9.16.16 Operation: grOBJECT_MOVE… grACTIVATE_GRID

Condor Operationen:
grOBJECT_MOVE,
grOBJECT_SCALE,
grSET_GRID,
grOBJECT_ACTIVATE,
grOBJECT_ROTATE,
grPROPERTIES,
grEDIT_CUT_DRW,
grEDIT_COPY_DRW,
grEDIT_PASTE_DRW,
grEDIT_UNDO_DRW,
grEDIT_REDO_DRW,
grACTIVATE_GRID

Man beachte auch die Wirkungsweise von `processGraphicChangeToDb`
<SOURCE> kann als Bedingung verwendet werden. ab 4.9.10

ab 4.10.6 Die Operation kann auch ein Bind Objekt enthalten vgl. *9.16.15*

9.16.17 Operation: grRotateMode S

ab 4.10.3 Umschalten zwischen 2 Koordinatensystemen. Der Umschalter kann gedrückt sein (Down) oder nicht (Up). Für beide Zustände wird definiert, ob für die Flächenobjekte (space) eine Drehung ausgeführt wird (Attribut bisG_NullRot) oder nicht. Ebenso wird dies für die Pläne definiert (Attribut Z_Drehwinkel). Der Initialzustand kann auch definiert werden.

Beispiel 1:

```
<Operation ID="G_Rotate" opCode="grRotateMode" forGraphic="Grafik"
           initialDown ="false" spaceRotateDown="true" spaceRotateUp="false"
           planRotateDown="true"  planRotateUp="false"/>
```

Beim Öffnen ist der Knopf nicht gedrückt.
Nicht gedrückt bedeutet: Die (Space) Objekte werden nicht gedreht dargestellt. Ebenso werden die Hintergrundspläne 1:1 dargestellt, also beides in Weltkoordinaten.
Gedrückt bedeutet: Die (Space) Objekte werden gemäss dem Attribut bisG_NullRot gedreht. Die Hintergrundspläne gemäss dem Attribut Z_Drehwinkel.

In diesem Fall sind Pläne und Objekt in Weltkoordinaten und können durch den Befehl in ein lokales Koordinatensystem gedreht werden. Im Modell ist dabei am Planattribut Z_Drehwinkel eine Methode zu hinterlegen, welche den das Attribut bisG_NullRot vom Stockwerk übernimmt.

Beispiel 2: Wenn der Plan in einem lokalen Koordinatensystem gezeichnet ist:

```
<Operation ID="G_Rotate" opCode="grRotateMode" forGraphic="Grafik"
           initialDown="true" spaceRotateDown="false" spaceRotateUp="true"
           planRotateDown="true"  planRotateUp="false"/>
```

Mit Down wird das globale Koordinatensystem angezeigt, dazu muss der Plan in die Gegenrichtung gedreht werden. Die Lesemethode am Attribut Z_Drehwinkel des Plans gibt den negierten Wert von bisG_NullRot des Stockwerks. Im Zustand Up wird der Plan 1:1 gezeigt (der ist ja lokal) und die (Space) Objekte werden ins lokale System gedreht (bisG_NullRot).

9.17 Custom Propagates

Die Grafik versendet die folgenden ‚custom propagates':

Custom Name	Beschreibung
GraphicLegend	Legende der Grafik in `additional.Data` in einem Variant-Array verpackt: Farbe, Text, Farbe, Text, etc. Das Application Element `<GraphicLegend>` versteht dieses Propagate.

Die Grafik versteht die folgenden ‚custom propagates':

Custom Name	Beschreibung
bisG_FAInstance	Aktuelle Färbung. Das Objekt muss eine Färbung sein.
PickBoxSize ab 4.10.7	Halbe Seitenlänge des Pick-Quadrates. Das PickQuadrat wird im Befehl CallCommand ‚Pick' vgl. Beispiel in ByronBIS_FormularReferenz Als Wert muss eine ganze positive Zahl übergeben werden. Mit -1 wird die Grösse abgefragt, d.h. die Grafik sendet wiederum ein custom propagate ‚PickBoxSize', das wiederum an das Formular gelenkt werden kann. Die Initialgrösse kann im Attribut pickBox definiert werden
bisG_GraphicView oder bisG_GraphicView:xy	Aktuelle Ansicht. Das Objekt muss eine Grfikansicht (bisG_GraphicView) sein. Alternativ (prioritär) kann nach einem Trennzeichen der Name der Ansicht angegeben werden, hier xy. Dieser Name wird mit den Namen der Datenbankansichten und mit den Namen der konfigurierten GraphicView verglichen.

Ein Custom Propagate kann zum Beispiel von der Operation opPropagateSubmenu genutzt werden:

```
<Operation ID="G_FarbSubMenu" opCode="opPropagateSubmenu" toElement="Graphic"
           propagateType="custom"       propagateName="bisG_FAInstance"
           caption="@bisG_FAInstance@"  icon="bisG_FAInstance">
    <ToObjects>START INSTANCES bisG_FAInstance</ToObjects>
</Operation>
```

Oder als 'normales' Propagate, Beispiel bisG_GraphicView:

```
<Propagate operation="custom" target="Graphic" nameTarget="bisG_GraphicView">
    <Condition>VALUE Name 'A*'</Condition>
    <Transform>
        START INSTANCES bisG_GraphicView VALUE Name 'Arch*'
    </Transform>
</Propagate>
```

```
<Propagate operation="custom" target="Graphic" nameTarget="bisG_GraphicView:std">
    <Condition>VALUE Name 'B*'</Condition>
</Propagate>
```

Im ersten Propagate wird eine Ansicht Arch… eingestellt, wenn das gewählte Objekt mit einem ‚A' beginnt.

Im zweiten Propagate wird die Ansicht mit Namen ‚std' eingestellt, wenn das gewählte Objekt mit einem ‚B' beginnt. Man beachte dass bei den Grafikansichten, welche im Tool konfiguriert werden der Name, und die Caption nicht gleich sind:

```
<GraphicView caption="Standard Info" name="std">
```

10 VdrawGraphic (>= 5.0.0)

<VdrawGraphic> stellt Pläne dar ähnlich wie <Graphic>. Es können mehrere Stockwerke eines Gebäudes gleichzeitig in 3D dargestellt werden. Die Darstellung erfolgt aufgrund der Attribute:

- Symbol an einer Position.
  zwingende Attribute: Position, bisG_SymbolName, bisG_SymbolVdrawGraphic (und Symbol muss existieren).
  weitere Attribute: Z_Drehwinkel, bisG_Scale, bisG_YStretch
- Flächen. zwingende Attribute: bisB_AreaGeometrie
- Verbindungslinien. (erst ab Byron/BIS v4.11.3)
  zwingende Attribute: bisB_Jointer
  weitere Attribute: bisB_JointType, bisG_Width, bis_Color.

Spezielle Attribute von VdrawGraphic:

```
<VdrawGraphic panel="pnGraphic"
         zoom="true"
         showHints="true"
         showMin="5.0"
         showScale="2.0"
         trueTypeScaling="true"
         priorityAttribute="MV_PickPriority"
         dropTopOnly = "true"
         processGraphicChangeToDb ="true"
         pickBox ="7"
         backgroundCache ="1"/>
```

`zoom` wenn true (default): nach dem Laden wird immer eine Operation 'Zoom Alles' ausgeführt. Wenn false: der Ausschnit wird nach dem erneuten Laden unverändert belassen.

`showHints` (default false) wenn true, dann wird in einem Tool-Tip Fenster das Objekt angezeigt, über dem sich der Zeiger befindet.

`showMin` (default 5.0) Minimale Grösse der Box, die verwendet wird beim Hinzoomen in der Operation zeigen. Beispiel: Wenn ein Symbol 20 cm gross ist, dann wird im default (5.0) ein Fenster von 5x5 Meter angezeigt.

`showScale` (default 2.0) Dies betrifft die Operation zeigen. Um diesen Faktor wird das Zoomfenster gegenüber dem Objekt bestimmt. Je grösser, desto weiter weg (kleiner) ist das Objekt. Mit 1.0 wird ganz auf das Objekt gezoomt.

`dropTopOnly` vgl Element Layer

Mit `trueTypeScaling` kann die TrueType-Skalierung an- oder ausgeschaltet werden. Default ist `true`. Die Alternative ist stufenlose Skalierung der Schrift als Vektorgrafik.

`priorityAttribute` und `dropTopOnly` vgl Element Layer.

Der Einsatz von `<VdrawGraphic>` innerhalb eines `<Tabbed>` kann Probleme verursachen, wenn die Umschaltlaschen ausgeblendet werden (NoTabsOnRuntime). Man sollte mit Grafik also immer `<Tabbed hideTabSet="false">` setzen. Oder man verwendet ein `<Pages>` anstelle eines `<Tabbed>`.

Mit `processGraphicChangeToDb` steuert man, ob eine Grafikänderung analysiert und in die Datenbank geschrieben werden soll. (default="true")

Mit `pickBox` wird die initale Grösse der Pickbox bestimmt. Dies ist der doppelte Wert welcher mit 'PickBoxSize' propagiert wird vgl. Dokumentation *Benutzerdefinierte Formulare*:

```
FormInformation.PropagateData('PickBoxSize', 0, -1, nil);
```
Dies wirkt sich also nur auf CallCommand aus.

Durch Einblenden der Toolbars:

```
<ShowToolBar status="true" menu="true"/>
```

Oder durch entsprechende Operationen, zB. grOBJECT_MOVE etc.

Mit `processGraphicChangeToDb="false"` können die Daten gegen Änderungen geschützt werden.

Mit `backgroundCache` wird definiert, wieviele Pläne beibehalten werden. Der Wert 1 (default) ermöglicht ein umschalten der Layer, ohne dass der Plan neu geladen werden muss. Ein grösserer Wert ermöglicht also ein schnelleres Umschalten zwischen verschiedenen Plänen wenn sie einmal geladen sind, braucht aber auch entsprechend mehr Speicher.

10.1 Element: Condition

Condition enthält die Bedingung in Form einer Filternavigation, ob ein load durchgeführt werden soll. Wenn das Eingangsobjekt die Bedingung nicht erfüllt (Ausgangsmenge leer), dann wird das Bestehende beibehalten.

Beispiel: wenn ein Gebäude als Eingangsobjekt auftritt, dann wird dieses dargestellt. Bei allen übrigen Objekten geschieht nichts.

```
<Condition>CLASS Building</Condition>
```

10.2 Element: Contents

Navigation zu einem oder mehreren Stockwerken.

Beispiel

```
<Contents>
    CLASS Building VIA Object_To_Children CLASS Floor
</Contents>
```

10.3 SubContents

Navigation von den Contents zu den Räumen.

`floorElevationAttr` muss ein Integer- oder Double-Attribut definieren, von dem die Reihenfolge und der Abstand der Stockwerke ausgelesen wird.

`floorLabelEvalEx` wird evaluiert, um eine Bezeichnung der Stockwerke zu erhalten, die dann angezeigt wird.

Mit `stretchFactor` kann ein Streckungsfaktor definiert werden, um den die Stockwerke weiter auseinander gezeichnet werden. Default: 1.0

Mit `shade` (>= 5.0.1) kann man bestimmen, wie sehr inaktive Ebenen abgedunkelt warden. Ein Wert zwischen 0 und 1 wird erwartet. Default: 0.67

Mit `transparency` (>= 5.0.1) kann man bestimmen, wie durchsichtig die oberste(active) Ebene ist. Ein Wert zwischen 0 und 1 wird erwartet, wobei 0 für undurchsichtig steht. Default: 0

Mit `topBgPlanOnly` (>= 5.0.1) kann man definieren, ob nur der Hintergrundsplan der aktiven Ebene gezeichnete wird oder alle. Default: `true`

Mit `layerSpread` (>= 5.0.1) kann man definieren, wie viel Platz des Zwischen-Stockwerk-Abstands Ebenen einnehmen. Bei mehreren Ebenen innerhalb eines Stockwerks wird der Z-Index einer Ebene auf die z-Achse in der 3d Grafik umgerechnet. Es wird ein Wert zwischen 0 und 1 erwartet, der als Faktor des Zwischen-Stockwerk-Abstands zu verstehen ist. Default ist 0.1 (10% des Zwischen-Stockwerk-Abstands). Bei kleinen Werten können Ebenen "verschmelzen" oder unschön gezeichnet werden. Bei grossen Werten ist je nach Blickwinkel gut sichbar, dass die Ebenen nicht auf gleicher Höhe liegen.

Beispiel

```
<SubContents stretchFactor="0.5" floorElevationAttr="bisB_FloorLevel" floorLabelEva-
lEx="$Name$" shade=”0.67” transparency=”0.0” topBgPlanOnly=”true” layerSpread=”0.1”>
    CLASS Floor [ OR VIA Object_To_Children] CLASS Room
</SubContents>
```

10.4 Element PaperColor

Bestimmt die Farben des Papiers.

```
<PaperColor inside="#FFFFFF"/>
```

10.5 Element Pick

Pickbarkeit der Fächen (shapes) und Positionskomponenten (symbols). Mit solid wird definiert, ob die Flächen beim Markieren rot werden, oder ob nur die Umrandung gefärbt wird.Note: s

```
<Pick shapes="false" symbols="true" solid="false"/>
```

Note: symbols / solids sind obsolet da dies mit dem Element Layer definiert werden können.

```
<Pick shapes="false"/>
```

```
<Layer shapes="false"/>
```

10.6 Element Layer

Damit werden Layer definiert. Jedes Objekt liegt auf genau einem Layer. Diese haben zwei Funktionen:
1. Die Pickbarkeit wird über den Layer gesteuert. (vgl. Operation `vgrPickable`)
2. Der Z-level der Objekte definiert sich über die priority des Layers auf welchem das Objekt liegt. Objekte mit höherer Priorität liegen über den Objekten mit niederer priority. Neben der Sichtbarkeit hat dies einen Einfluss bei der Bestimmung des Drop-Targets. Mit `dropTopOnly` steuert man ob man nur das Objekt mit der höchsten Priorität (true) oder alle (false) haben möchte. Im letzten Fall kann man in der Target Filternavigation auch `SORT (-graphicPriority) PICK 0 1` einbauen.

Die Priorität der Objekte bestimmt sich aus dem Integer-Attribut, welches in priorityAttribute am Element VdrawGraphic angegeben ist. Bsp, wenn demo_graphicPriority ein integerAttribut ist:

```
<VdrawGraphic priorityAttribute="demo_graphicPriority" dropTopOnly="true" />
```

Wenn kein Wert am Objekt definiert ist, gilt der Default:

Für Flächenobjekte:	200
Für Verbindungsobjekte:	300
Für Positionsobjekte:	400
Beschriftungen:	800

Pickbarkeit der Fächen (shapes) und Positionskomponenten (symbols). Mit solid wird definiert, ob die Flächen beim Markieren rot werden, oder ob nur die Umrandung gefärbt wird.

```
<Layer name="BisLayerFloor" priority="200" />
<Layer name="BisLayerRoom" priority="201" />
<Layer name="BisLayerZone" priority="202" />
```

In diesem Beispiel müsste an der Klasse Room das Attribut `demo_graphicPriority` definiert sein, mit einem Vorgabewert von 201.

Das Attribut `name` kann weggelassen werden. Der Name des Layers hat in der Konfiguration keine Bedeutung und ist höchstens bei einem Export ersichtlich.

Wenn der Benutzer mehrere Objekte pickt, dann werden nur die Objekte mit der grössten Priorität gewählt. zB. wenn eine Komponente in einem Raum liegt und der Benutzer waählt die Komponente und damit auch den darunter liegenden Raum, dann wird der Raum nicht selektiert, da er einen niedrigere Priorität hat. Dieses Verhalten kann man mit der Alt-Taste aufheben. So kann man zB. alle Objekte in einem Rechteck wählen, indem man eine Ecke wählt, das Rechteck aufzieht und vor dem Loslassen der Maustaste noch die Alt-Taste drückt. Das funktioniert auch in Kombination mit additiver Selektion (Ctrl-Taste)

10.7 Element ShowToolBar

Definiert, welche VectorDraw-Toolbars eingeblendet werden. Default alle false

```
<ShowToolBar status="true" menu="true" layoutTab="false" />
```

10.8 Element PlanToObject / ObjectToPlan

Diese zwei Elemente sind nur von Bedeutung, wenn an den Gebäuden / Stockwerken ein Koordinatensystem definiert ist (bisG_NullX, bisG_NullY, bisG_NullRot). Das Koordinatensystem wird vom Layout-Editor wie folgt interpretiert: Die Objekte (Räume, Komponenten) werden um diese Transformation ‚zurück' verschoben/gedreht und passen dann zum ‚lokalen' Architekturplan. Die Objekte sind in globalen Koordinaten abgelegt.

Dieses Verhalten erreicht man durch Definition des gewünschten Koordinatensystems:

```
<ObjectToPlan>
    RECALL OBJ_StartObjects CLASS Space
</ObjectToPlan>
```

Wenn nichts definiert ist, dann wird das StartObjekt genommen, sofern es genau eines gibt. Damit erübrigt sich obige Definition im Allgemeinen. Um dieses default Verhalten zu unterdrücken kann zB. `<ObjectToPlan>BLOCK</ObjectToPlan>` geschrieben werden.

Möchte man Objekte aus verschiedenen Koordinatensystemen gleichzeitig anzeigen, dann muss obiger Mechanismus ausgeschaltet werden. Dies ist Explizit notwendig, wenn es genau ein Startobjekt gibt und dieses ein ‚falsches' Koordinatensystem besitzt. Möchte man nun dieselben (Stockwerk) Pläne verwenden, dann müssen diese zu den Objekten hin verschoben werden. Das System muss also wissen, zu welchem Koordinatensystem (Stockwerk / Gebäude) der Plan gehört. Dies kann wie folgt definiert werden:

```
<PlanToObject>
    VIA Doc_Referenced_By CLASS Space
</PlanToObject>
```

Dies entspricht der Standard-Modellierung in Byron/BIS

10.9 Element GraphicViews

Definiert die Ansicht (dargestellte Objekte, Farbe, Hintergrundsplan und Beschriftung).

Bestehende Grafikansichten können mit <Use> verwendet werden. Mit <GraphicView> kann auch eine Ansicht ad-hoc definiert werden.

Ansichten können durch den Benutzer umgestellt werden (userinetrface=true) oder durch ein Custom-Propagate. Die `caption` wird in der Combo-Box angezeigt, der `name` wird vom Custom-Propagate verwendet.

NavContents definiert die darzustellenden Objekte. Das Eingangsobjekt wird durch inputIs definiert:

nothing:	die Navigation wird gar nicht ausgeführt, es kommen keine zusätzlichen Objekte durch die Navigation.
showedObjects:	alle Objekte, welche durch Start -> Contents gezeichnet werden konnten.
navigatedObjects:	alle Start -> Contents Objekte
startObject:	nur das Startobjekt.

Plan definiert die Hintergrundspläne. Ausgangsobjekt der Filternavigation ist das Startobjekt (z.B. Floor). Wenn das erreichte Objekt ein Tool ist, dann muss es ein Layout-Editor sein und der Hintergrund wird aus dem Dokument gelesen (Layout-Editor – Hintergrund). Wenn das Objekt ein Doc_File ist, dann wird der Hintergrund aus der Datei gelesen. Wenn für minSizeForThread ein Wert grösser als 0 angegeben wird, dann erfolgt das Lesen in einem separaten Thread (im Hintergrund parallel zur sonstigen Verarbeitung).

Label definiert die Grafik-Beschriftung (bisG_TextContainer).

Color definiert die Färbung (bisG_FAInstance). Mit dem Attribut fix kann direkt eine Farbe definiert werden, dies anstelle einer Filternavigation zu einer Färbung.
<Color fix="#1010E0"/>

Hatch definiert den Stil der Raumgeometriedarstellung. Style = solid, line oder hollow. (Default ist: `<Hatch style="solid"/>`)

solid:	die Polygone werden gefüllt dargestellt.
hollow:	es wird nur die Umrandung gezeichnet, auch bei der Selektion.
line:	es wird schraffiert gezeichnet. In diesem Fall beinhaltet Hatch noch ein Element Line.

Line definiert den Linienstil im Fall `style="line" mit den Attributen width= Dicke, widthMode=0` Dicke in Pixel, `widthMode=1` Dicke in mm Papier, `widthMode=2` Dicke in Meter (Welt). `angle=` Winkel der Schraffur in Grad

Beispiel, welches auch den default Werten entspricht:

```
<Hatch style="line">
  <Line width="1" widthMode="0" distance="0.15" angle="45" />
</Hatch>
```

```
<GraphicViews comboWidth="290" userinterface="true">
    <Use>START INSTANCES bisG_GraphicView</Use>
    <GraphicView caption="Standard Info" name="std">
        <NavContents inputIs="showedObjects">
            VIA "Room_To_Component"
        </NavContents>
        <Plan minSizeForThread="20">
            VIA Doc_Documents
            CLASS Doc_File
            VALUE Doc_Path "*GRUNDRISS.drw"
        </Plan>
        <Label>
            START INSTANCES bisG_TextContainer VALUE Name 'Rauminformation'
        </Label>
        <Color>
            START INSTANCES bisG_FAInstance VALUE Name 'DIN*'
        </Color>
    </GraphicView>
</GraphicViews>
```

Ansichten können mit `<SubGraphicView>` in Untermenüs gruppiert werden ab 4.9.11

```
<GraphicViews comboWidth="290" userinterface="true">
    <GraphicView caption="Standard Info" name="std"> … </GraphicView>
    <SubGraphicView caption="Standard Info">
        <Use>START INSTANCES bisG_GraphicView VALUE Name "Raum*"</Use>
        <GraphicView caption="Standard Info" name="std"> … </GraphicView>
    </SubGraphicView>
</GraphicViews>
```

10.10 Grafik Operationen

Die Grafik-Operationen unterstützen ein forVdrawGraphic, damit kann der Befehl unabhängig vom Fokus.

Beispiel:

```
<Operation ID="grAreaPick" opCode=„vgrAreaPickable" forVdrawGraphic="Graphic"/>
```

S: Schalterbefehl, d.h. der Befehlt schaltet die Einstellung um (angehakt <-> nicht angehakt)

A: Befehl der weitere Eingaben erfordert. Diese Befehle können mit dem Befehl grCancel abgebrochen (A) werden.

D&D Drag/Drop Befehl

10.10.1 Operation: vgrCancel

Ein A Befehl wird abgebrochen.

10.10.2 Operation S: vgrAreaPickable

Shapes, das sind Flächendarstellungen von Räumen sind pickbar, bzw. nicht pickbar. Die Operation ist ein Umschalter (checked Haken).
obsolet vgl. grPickable.

```
<Operation ID="x" opCode=„vgrPickable" forVdrawGraphic="Graphic" priorityLayer="200"
           icon="Space"/>
```

10.10.3 Operation S: vgrSymbolPickable

Blöcke, das sind Symboldarstellungen von Komponenten sind pickbar, bzw. nicht pickbar. Die Operation ist ein Umschalter (checked Haken).
obsolet vgl. grPickable

```
<Operation ID="x" opCode=„vgrPickable" forVdrawGraphic="Graphic" priorityLayer="400"
           icon="Component"/>
```

10.10.4 Operation S: vgrPickable

Objekte von einem bestimmetn Layer pickbar schalten vgl. Element Layer. Der Layer wird durch seine priority identifiziert (`priorityLayer=`). Mehrere Layer können durch ; getrennt angegeben werden oder ganze Bereiche mit -
Bsp "150;152;160-165;170;180-199"
Der Initialwert kann durch pickable=<boolean> gesetzt werden (default = "true").

```
<Operation ID="x" opCode=„vgrPickable" forVdrawGraphic="Graphic" priorityLayer="195-205"
           icon="Space"/>
```

10.10.5 Operation S: vgrSolidHighlight

Sollte nicht verwendet werden. Default ist «true», d.h. selektierte Flächen werden durch das Einfärben der Fläche hervorgehoben. Nur Umrandung sit oft durch den Hintergrundplan verdeckt.

initialDown: Steuert den Anfangszustand des Schalters (gedrückt bzw. ungedrückt).
Default: "true"

10.10.6 Operation: vgrPAN

Der Ausschnitt kann verschoben werden. ab Version 5.4.5

10.10.7 Operation: vgrZoomPage

Der Ausschnitt wird auf das Blatt gesetzt.

10.10.8 Operation: vgrZoomTotal

Der Ausschnitt wird so gesetzt, dass alle Objekte sichtbar sind.

10.10.9 Operation A: vgrZoomDetail

Der Zoom-Befehl wird initiiert. Es wird ein Fadenkreuz angezeigt, damit der Benutzer den ersten Eckpunkt eingeben kann. Danach folgt ein Rechteck um den Ausschnitt auszuwählen.

10.10.10 Operation D&D: vgrDropPosition

Setzt das Attribut Position auf die Koordinate des Drop Punktes. Wenn die Operation noch ein <Bind> enthält, dann wird noch eine Assoziation auf die Fläche hergestellt. <Source>, <Target> und ‚forVdrawGraphic' werden ebenfalls unterstützt.

Es können auch mehrer Objekte gleichzeitig verschoben werden. Dieses verhalten kann auch durch den mode gesteuert werden `mode = "positionOnly", "moveOnly", "both"`
Default `"both"`: Damit wird mit einem Objekt positioniert und bei mehreren Objekten verschoben.
Beispiel:

```
<Operation ID="G_DropPosition" opCode=„vgrDropPosition"
           caption="grDropPosition_Caption" forVdrawGraphic="Graphic">
    <Bind assoc="Room_Of_Component"/>
</Operation>
```

10.10.11 Operation: vgrMeasure

Mite dieser Operation könne Messungen auf dem Plan durchgeführt werden.

`moveFormat`: Definiert den Text während des Messens. Es stehen folgende Werte zur Verfügung:
%0: Distanz
%1: Dx gemäss Ansicht
%2: Dy gemäss Ansicht
Default: `moveFormat="Distanz: %0:.2f m"`

Beispiel:

```
<Operation ID="G_Measure" opCode="vgrMeasure"" forVdrawGraphic="Graphic"
            moveFormat="%0:.2f m  (%1:.2f  %2:.2f)" >
    <Source> CLASS Position_Component </Source>
</Operation>
```

10.10.12 Operation: vgrMove

Damit kann das Attribut ‘Position’, verändert werden.

Beispiel:

```
<Operation ID="G_Moven" opCode="vgrMove"" forVdrawGraphic="Graphic">
    <Source> CLASS Position_Component </Source>
</Operation>
```

10.10.13 Operation: vgrPrint

Grafik Drucken. Der Befehl ist im Kapitel Operation (vgrPrint) beschrieben.

10.10.14 Operation: vgrOBJECT_ACTIVATE…

vgrOBJECT_ACTIVATE ermöglicht das Bearbeiten der Punkte einer Fläche direkt in der Grafik (ohne zusätzlichen Dialog)

Man beachte auch die Wirkungsweise von `processGraphicChangeToDb`
<SOURCE> kann als Bedingung verwendet werden.

Die Operation kann auch ein Bind Objekt enthalten vgl. *9.16.15*

10.10.15 Operation: vgrRotateMode S

Umschalten zwischen 2 Koordinatensystemen. Der Umschalter kann gedrückt sein (Down) oder nicht (Up). Für beide Zustände wird definiert, ob für die Flächenobjekte (space) eine Drehung ausgeführt wird (Attribut bisG_NullRot) oder nicht. Ebenso wird dies für die Pläne definiert (Attribut Z_Drehwinkel). Der Initialzustand kann auch definiert werden.

Beispiel 1:

```
<Operation ID="G_Rotate" opCode=„vgrRotateMode" forVdrawGraphic="Grafik"
           initialDown ="false" spaceRotateDown="true" spaceRotateUp="false"
           planRotateDown="true"  planRotateUp="false"/>
```

Beim Öffnen ist der Knopf nicht gedrückt.
Nicht gedrückt bedeutet: Die (Space) Objekte werden nicht gedreht dargestellt. Ebenso werden die Hintergrundspläne 1:1 dargestellt, also beides in Weltkoordinaten.
Gedrückt bedeutet: Die (Space) Objekte werden gemäss dem Attribut bisG_NullRot gedreht. Die Hintergrundspläne gemäss dem Attribut Z_Drehwinkel.

In diesem Fall sind Pläne und Objekt in Weltkoordinaten und können durch den Befehl in ein lokales Koordinatensystem gedreht werden. Im Modell ist dabei am Planattribut Z_Drehwinkel eine Methode zu hinterlegen, welche den das Attribut bisG_NullRot vom Stockwerk übernimmt.

Beispiel 2: Wenn der Plan in einem lokalen Koordinatensystem gezeichnet ist:

```
<Operation ID="G_Rotate" opCode=„vgrRotateMode" forVdrawGraphic="Grafik"
           initialDown="true" spaceRotateDown="false" spaceRotateUp="true"
           planRotateDown="true"  planRotateUp="false"/>
```

Mit Down wird das globale Koordinatensystem angezeigt, dazu muss der Plan in die Gegenrichtung gedreht werden. Die Lesemethode am Attribut Z_Drehwinkel des Plans gibt den negierten Wert von bisG_NullRot des Stockwerks. Im Zustand Up wird der Plan 1:1 gezeigt (der ist ja lokal) und die (Space) Objekte werden ins lokale System gedreht (bisG_NullRot).

10.10.16 Operation: vgrELEVATOR

Mit dieser Operation kann der Fokus zwischen den geladenen Stockwerken zyklisch gewechselt werden.

10.10.17 Operation: vgrROTATE S (>= 5.0.1)

Mit dieser Operation wird in den Rotationsmodus umgeschaltet. Somit kann man die Grafik 3-dimensional rotieren.

10.10.18 Operation: vgrPERSPECTIVE (>= 5.0.1)

Mit dieser Operaion kann man zwischen paralleler und perspektivischer 3d-Ansicht gewechselt werden.

10.10.19 Operation: vgrViewParam (>= 5.0.2)

Mit dieser Operaion kann man die Kameraposition an einen beliebigen Punkt setzen. Erwartet werden azimiuth, tilt, twist. Die Winkel werden in Grad angfegeben:

`azimuth`: Winkel in der X-Y Ebene, von der X-Achse aus im Gegenuhrzeigersinn. Default = -90, d.h. in Richtung negative Y-Achse.

`tilt:` Erhebungswinkel zur X-Y Ebene. Default 90, d.h. aus Richtung z-Achse.

`twist:` Kippwinkel der Kamera . Default = 0, d.h. 'gerade'

10.10.20 Operation: vgrSetRender (>= 5.0.2)

Mit dieser Operaion kann man vordefinierte VectorDraw Rendereigenschaften sezten. Erwartet wird ein String. Mögliche Werte sind VTOP, VBOTTOM, VLEFT, VRIGHT, VFRONT, VBACK, VISW, VISE, VINE, VINW, RENDER, VROT, SHADE, SHADEON, HIDE, WIRE. Siehe auch in der VectorDraw Hilfe die View3D-Funktion.

```
<Operation ID="G_SetRender" opCode="vgrSetRender" render="HIDE" forVdrawGraphic="gra-
phic3D"/>
```

10.11 Custom Propagates

Die Grafik versendet die folgenden ‚custom propagates':

Custom Name	Beschreibung
GraphicLegend	Legende der Grafik in `additional.Data` in einem Variant-Array verpackt: Farbe, Text, Farbe, Text, etc. Das Application Element `<GraphicLegend>` versteht dieses Propagate.

Die Grafik versteht die folgenden ‚custom propagates':

Custom Name	Beschreibung
bisG_FAInstance	Aktuelle Färbung. Das Objekt muss eine Färbung sein.
bisG_GraphicView oder bisG_GraphicView:xy	Aktuelle Ansicht. Das Objekt muss eine Grfikansicht (bisG_GraphicView) sein. Alternativ (prioritär) kann nach einem Trennzeichen der Name der Ansicht angegeben werden, hier xy. Dieser Name wird mit den Namen der Datenbankansichten und mit den Namen der konfigurierten GraphicView verglichen.

Ein Custom Propagate kann zum Beispiel von der Operation opPropagateSubmenu genutzt werden:

```
<Operation ID="G_FarbSubMenu" opCode="opPropagateSubmenu" toElement="Graphic"
           propagateType="custom"       propagateName="bisG_FAInstance"
           caption="@bisG_FAInstance@"  icon="bisG_FAInstance">
    <ToObjects>START INSTANCES bisG_FAInstance</ToObjects>
</Operation>
```

Oder als 'normales' Propagate, Beispiel bisG_GraphicView:

```
<Propagate operation="custom" target="Graphic" nameTarget="bisG_GraphicView">
    <Condition>VALUE Name 'A*'</Condition>
    <Transform>
        START INSTANCES bisG_GraphicView VALUE Name 'Arch*'
    </Transform>
</Propagate>
```

```
<Propagate operation="custom" target="Graphic" nameTarget="bisG_GraphicView:std">
    <Condition>VALUE Name 'B*'</Condition>
</Propagate>
```

Im ersten Propagate wird eine Ansicht Arch… eingestellt, wenn das gewählte Objekt mit einem ‚A' beginnt.

Im zweiten Propagate wird die Ansicht mit Namen ‚std' eingestellt, wenn das gewählte Objekt mit einem ‚B' beginnt. Man beachte dass bei den Grafikansichten, welche im Tool konfiguriert werden der Name, und die Caption nicht gleich sind:

```
<GraphicView caption="Standard Info" name="std">
```

11 GraphicLegend

Mit dem Application Element `<GraphicLegend>` kann das Propagate "GraphicLegend" des Application Elements `<Graphic>` abgefangen werden und daraus ein Grid als Legende für die Grafik anzeigt werden.

11.1 Propagates

Die Grafik-Legende versteht folgende Propagates:

Custom Name	Beschreibung
GraphicLegend*	Farb- und Textinformationen um die Legende einer Grafik aufzubauen.

12 Form

Mit dem Application Element <Form> können Formulare dargestellt werden. Diese können auch spezielle Komponenten und Funktionen für das Empfangen und Versenden von Propagate enthalten.

12.1 Attribut: ignoreMultiInputObjects

`ignoreMultiInputObjects = <boolean>` wenn `true`, dann werden mehrere Eingabeobjekte ognoriert (es wird nur das erste Objekt verwendet. Wenn `false`, wird bei mehreren Eingabeobjekten das Formular entfernt und eine Fehlermeldung angezeigt.
Default ist `false`.

12.2 Attribut: keepLastForm

`keepLastForm = <boolean>` wenn `true`, dann wird das letzte verwendete Formular beibehalten, auch wenn keine oder mehrere Eingabeobjekte vorhanden sind oder kein Formular für das Eingabeobjekt gefunden wird. Wenn `false`, wird das Formular geschlossen, falls keine oder mehrere Eingabeobjekte vorhanden sind oder kein Formular für das Eingabeobjekt gefunden wird.
Default ist `true`.

12.3 Element: TextNoObject

Definiert den Text für "Kein Objekt gewählt". Als Default wird die Text-ID "Frmwrk_FormTextNoObject" verwendet.

12.4 Element: TextMoreObjects

Definiert den Text für "Mehr als ein Objekt gewählt". Als Default wird die Text-ID "Frmwrk_FormTextMoreObjects" verwendet.

12.5 Element: TextNoForm

Definiert den Text für "Kein registriertes Formular gefunden!". Als Default wird die Text-ID "Frmwrk_FormTextNoForm" verwendet.

12.6 Element: FormContents

Falls `name` nicht angegeben wurde, wird in FormContents eine FilterNavigation angegeben, die definiert welches Objekt im Formular angezeigt wird. **Bis v4.6.5** wurde dann das für das Objekt registrierte Formular angezeigt. **Ab v4.6.5** wird zusätzlich überprüft, ob das gefundene Objekt ein Formular ist. Wenn es ein Formular ist, wird dieses angezeigt und das **erste** der an das Applikationselement übergebenen Objekte (per Propagate oder StartObjects) wird als DataSource vewendet.

Beispiel: Ein Formular propagiert eine Person und eine Funktion an das Form-Applikationselement. Die FormContents FilterNavigation navigiert nun über die Assoziation "bisB_FunctionToForm" – es wird ein anderes Formularobjekt gefunden. Das gefundene Formularobjekt wird angezeigt, mit der propagierten Person als DataSource. Wenn im Propagate zuerst die Funktion und dann die Person propagiert worden wäre, würde die Funktion als DataSource genommen.

Hinweis: Ab Byron/BIS v4.6.4 ist die Angabe des Elements `FormContents` optional. Ohne Angabe von FormContents werden *alle* Objekte mit dem *registrierten* Formular angezeigt.

12.6.1 Attribut: name

`name = <Text>` definiert den Formularnamen (BIS-Attribut "Name"), welcher verwendet werden soll (fixe Formulare).

Beispiel 1 (nur Objekte der Klasse `Space` anzeigen; registriertes Formular verwenden)

```
<Form ID="Form1" panel="right">
    <FormContents>CLASS Space</FormContents>
    <Propagate operation="load" target="Form2" name="tPropSelect"/>
</Form>
```

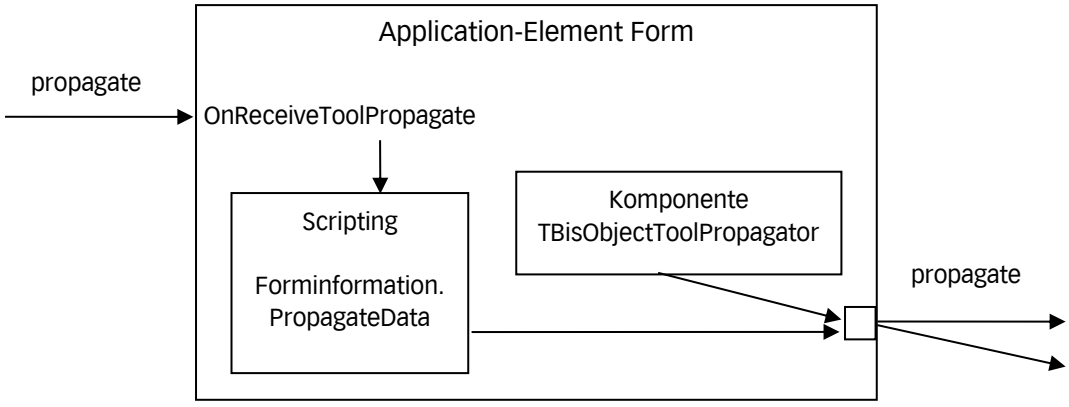
Beispiel 2 (alle Objekte anzeigen; immer das Formular "Building" verwenden)

```
<Form ID="Form1" panel="right">
    <FormContents name="Building"/>
    <Propagate operation="load" target="Form2" name="tPropSelect"/>
</Form>
```

12.7 Propagate in Formularen

Die propagate welche im Application Element definiert sind (Ausgang), werden durch die Komponente **TBisObjectToolPropagator** bedient. Dabei wird der Name der Komponente als name des propagate verwendet. Eine weitere Möglichkeit bietet der Aufruf von **procedure TFormInformation.PropagateData** im Scripting.

Empfangene propagate können im Ereignis OnReceiveToolPropagate der FormInformation erhalten werden. In jedem Fall wird bei einem load BisDataSource mit dem Objekt geladen. Dies ermöglicht die Verwendung von ‚gewöhnlichen' Formularen.



13 PropertyList

Mit dem Application Element <PropertyList> kann das Eigenschaftsblatt dargestellt werden.

Es könne Attribute und Assoziationen angezeigt werden.

13.1 Element: DisplayOptions

Das Element <DisplayOptions> enthält die Optionen zur Konfiguration der Anzeige des Eigenschaftsblattes.

13.1.1 Attribute von DisplayOptions

- `showHeader = <boolean>` wenn `true`, dann werden die Spaltenüberschriften angezeigt.
  Default ist `false`.
- `headerDescription = <text>` definiert den Text für die Spalte "Eigenschaft".
  Default ist "Eigenschaft".
- `headerValue = <text>` definiert den Text für die Spalte "Wert".
  Default ist "Wert".
- `headerUnit = <text>` definiert den Text für die Spalte "Einheit".
  Default ist "Einheit".
- `showUnits = <boolean>` wenn `true`, dann wird die Spalte "Einheit" angezeigt.
  Default ist `false`.
- `useDisabledColor = <boolean>` wenn `true`, dann wird die Farbe der Eigenschaft "DisabledColor" für den Hintergrund verwendet.
  Default ist `true`.
- `showHidden = <boolean>` wenn `true`, dann werden auch versteckte Attribute/Assoziationen angezeigt.
  Default ist `false`.
- `showReadOnly = <boolean>` wenn `true`, dann werden auch schreibgeschützte Attribute/Assoziationen angezeigt.
  Default ist `true`.
- `showReadWrite = <boolean>` wenn `true`, dann werden auch Attribute/Assoziationen mit Lese-/Schreibzugriff angezeigt.
  Default ist `true`.
- `showOnlyExisting = <boolean>` wenn `true`, dann werden nur Attribute/Assoziationen angezeigt, welche gesetzt sind (existieren).
  Default ist `false`.
- `showAssociations = <boolean>` wenn `true`, dann werden auch Assoziationen angezeigt.
  Default ist `true`.
- `showAttributes = <boolean>` wenn `true`, dann werden auch Attribute angezeigt.
  Default ist `true`.

13.2 Element: PropertyListOptions

Das Element <PropertyListOptions> enthält die Optionen zur Konfiguration der Änderungen im Eigenschaftsblatt.

13.2.1 Attribute von PropertyListOptions

- `allowDragAssociation = <boolean>` wenn `true`, dann können die Objekte einer Assoziation mit der Maus gezogen (Drag) werden.
  Default ist `false`.
- `allowDropAssociation = <boolean>` wenn `true`, dann kann ein Objekt auf eine "zu 1"-Assoziation losgelassen (Drop) werden.
  Default ist `false`.
- `allowRemoveAssociation = <boolean>` wenn `true`, dann das Objekt einer "zu 1"-Assoziation entfernt werden. Dies kann in dem Popupmenü über den Menüeintrag "Entfernen" aufgerufen werden.
  Default ist `false`.
- `allowDblClickOpenAssociation = <boolean>` wenn `true`, dann können die Objekte einer Assoziation mit einem Doppelklck geöffnet werden.
  Default ist `false`.
- `allowOpenAssociation = <boolean>` wenn `true`, dann können die Objekte einer Assoziation geöffnet werden. Dies kann in dem Popupmenü über den Menüeintrag "Öffnen" aufgerufen werden.
  Default ist `false`.
- `allowPropertiesAssociation = <boolean>` wenn `true`, dann kann das Eigenschaftsblatt für die Objekte einer Assoziation geöffnet werden. Dies kann in dem Popupmenü über den Menüeintrag "Eigenschaften" aufgerufen werden.
  Default ist `false`.
- `allowDeleteAttribute = <boolean>` wenn `true`, dann kann der Wert eines Attributs gelöscht werden. Dies kann in dem Popupmenü über den Menüeintrag "Attribut löschen" aufgerufen werden. (ab V 4.10.4)
  Default ist `false`.
- `allowSetDefaultAttribute = <boolean>` wenn `true`, dann kann der Wert eines Attributs auf den Vorgabewert gesetzt werden. Dies kann in dem Popupmenü über den Menüeintrag "Attribut auf Vorgabe" aufgerufen werden. (ab V 4.10.4)
  Default ist `false`.

13.3 Element: IncludeProperty

Um bestimmte Attribute oder Assoziationen anzuzeigen, wird das Element <IncludeProperty> verwendet. Für jede Property (Attribut oder Assoziation) wird ein eigenes Element definiert. Dabei sind auch Platzhalter (z.B. bisB_*) erlaubt.

13.4 Element: ExcludeProperty

Um bestimmte Attribute oder Assoziationen nicht anzuzeigen, wird das Element <ExcludeProperty> verwendet. Für jede Property (Attribut oder Assoziation) wird ein eigenes Element definiert. Dabei sind auch Platzhalter (z.B. bisT_*) erlaubt.

Beispiel 1

```
<PropertyList ID="PropertyList1" panel="RightBottom">
    <DisplayOptions showHeader="true"
                    headerDescription="Eigenschaft"
                    headerValue="Wert"
                    headerUnit="Einheit"
                    showUnits="true"
                    useDisabledColor="true"
                    showHidden="false"
                    showReadOnly="true"
                    showReadWrite="true"
                    showOnlyExisting="false"
                    showAssociations="true"
                    showAttributes="true"/>
    <PropertyListOptions allowDragAssociation="true"
                         allowDropAssociation="true"
                         allowRemoveAssociation="true"
                         allowDblClickOpenAssociation="true"
                         allowOpenAssociation="true"
                         allowPropertiesAssociation="true"
                         allowDeleteAttribute="true"
                         allowSetDefaultAttribute="true"/>
    <IncludeProperty>bisB_NetGroundArea</IncludeProperty>
    <IncludeProperty>bisG_NumberOfHoles</IncludeProperty>
    <IncludeProperty>bisB_NumberOfPersons</IncludeProperty>
    <IncludeProperty>bisB_AllPersonsIn</IncludeProperty>
    <IncludeProperty>bisC_ObjectToCostCenter</IncludeProperty>
    <Propagate operation="load" target="Form2" name="tPropSelect"/>
</PropertyList>
```

Beispiel 2

```
<PropertyList ID="PropertyList1" panel="RightBottom">
    <DisplayOptions showHeader="true"
                    headerDescription="Eigenschaft"
                    headerValue="Wert"
                    headerUnit="Einheit"
                    showUnits="true"
                    useDisabledColor="true"
                    showHidden="false"
                    showReadOnly="true"
                    showReadWrite="true"
                    showOnlyExisting="false"
                    showAssociations="true"
                    showAttributes="true"/>
    <PropertyListOptions allowDragAssociation="true"
                         allowDropAssociation="true"
                         allowRemoveAssociation="true"
                         allowDblClickOpenAssociation="true"
                         allowOpenAssociation="true"
                         allowPropertiesAssociation="true"/>
    <ExcludeProperty>bisC_*</ExcludeProperty>
    <Propagate operation="load" target="Form2" name="tPropSelect"/>
</PropertyList>
```

14 Planner

Mit dem Application Element <Planner> können Objekte in einem Kalender mit Tages- oder Wochenansichten dargestellt werden.

Schematischer Aufbau erstellen (Grafik mit Sidebar, Header, Item)

14.1 Beispiel

```
<Planner ID="planner" panel="pnPlanner" onDoubleClick="Open"
         sendSelectNotification="true" receiveSelectNotification="false"
         propagateOnFocusChanged="true">
    <Contents>
        [
            VIA bisP_MaintGroupToPerson CLASS Person SORT (+ bisB_Lastname)
        OR
            VIA Object_To_Children CLASS bisP_MaintGroup
        ]
    </Contents>
    <PlannerView name="xxx">
       <!-- ... -->
    </PlannerView>
    <PlannerRule>
        <!-- ... -->
    </PlannerRule>
    <PlannerHoliday>
        <!-- ... -->
    </PlannerHoliday>
</Planner>
```

14.2 Beschreibung

14.2.1 XML-Attribute

Bezeichnung	Beschreibung
allowItemDrop	`allowItemDrop = <boolean>` Mit allowItemDrop="true" wird auch das dargestellte Objekt (Item) als Ziel (Target) beim Drag&Drop erkannt. Optional. Default = "false"
backgroundSelection	`backgroundSelection = <boolean>` Mit backgroundSelection="true" wird auch der Hintergrund als Selektion erkannt. Als Hintergrundobjekt wird das jeweilige Spalten- bzw. Zeilenobjekt verwendet. Optional. Default = "false"
showHints	`showHints = <boolean>` Mit showHints="true" werden die Hints für die dargestellten Objekte (Items) angezeigt. Die Formatierung von Hints kann im Element innerhalb der <PlannerRule> definiert werden.

	Optional. Default = "false"

14.2.2 Ausgelöste propagates

Aktion	Operation
Datum setzen	Date

Beispiel:

```
<Propagate operation="custom" name="Date" target="calendar"/>
```

14.2.3 Empfangene propagates

Operation	Beschreibung
Date	Setzt das Datum im Kalender
ScrollIntoView	Scrollt zum mitgelieferten Objekt

Beispiel:

```
<Propagate operation="custom" name="Date" target="planner"/>
```

14.2.4 Parameter für die Filternavigation

Siehe auch: Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
OBJ_BackgroundSelection	Objekt	Selektiertes Spalten- bzw. Zeilenobjekt
VAL_CurrentDate	Datum	Aktuelles Datum des Kalenders
VAL_SelectedStartDate	Datum	Selektiertes Start-Datum des Kalenders
VAL_SelectedEndDate	Datum	Selektiertes End-Datum des Kalenders
VAL_SelectedDuration	Datum	Dauer der Selektion des Kalenders
VAL_FirstDate	Datum	Erstes Datum im Kalender
VAL_LastDate	Datum	Letztes Datum im Kalender

Der Parameter muss die Element-ID als Prefix beinhalten.

Beispiel:

```
<Planner ID="planner">
...
VALUE bisB_CreationDate = DATAOF 'planner.VAL_CurrentDate'
```

14.3 Element: PlannerHeaderCaption

Eine Formel welche für jede Zeilenüberschrift ausgeführt wird.

14.3.1 Beispiel

```
<Planner ID="planner" panel="pnPlanner">
    ...
    <PlannerHeaderCaption>$bisB_SpaceId$</PlannerHeaderCaption>
    ...
</Planner>
```

14.4 Element: PlannerView

Definiert eine oder mehrere Ansichten, welche verwendet werden sollen.

14.4.1 Beispiel

```
<PlannerView name="Tag" orientation="vertical">
    <Timeline shownDays="1" startingDay="0" resolution="30"/>
    <Header width="100"/>
    <Clipping visible="true" before="7" after="19"/>
    <Navigation step="1"/>
</PlannerView>
```

14.4.2 Attribute

Bezeichnung	Beschreibung
name	Bezeichnung der PlannerView, wird in der Auswahlliste angezeigt Pflichtfeld
default	 Werte: true, false Vorgabewert: false
orientation	Ausrichtung Werte: horizontal, vertical Pflichtfeld

14.4.3 Elemente

Bezeichnung	Beschreibung
Clipping	Dient dazu, Randzeiten auszublenden (früh morgens und spät abends). Attribute: • after: Uhrzeit (genauer Stunde), **vor** welcher keine Termine angezeigt werden. Vorgabe: 0 • before: Uhrzeit (genauer Stunde), **nach** welcher keine Termine angezeigt werden. Vorgabe: 0 • visible: Wenn false, dann werden die Randzeiten nicht angezeigt. Wenn true und die Eigenschaft *showndays* des Elements *TimeLine* (siehe unten) ist = „1“, dann werden die Randzeiten farblich hinterlegt. Vorgabe: true

	N.B: Die Bedeutung von after und before wurde in Version **v4.11.8** sowie in der Nachproduktion **v4.11.7.2208** geändert (A-004751). **N.B**.: Die in *after* und *before* verwendeten Uhrzeiten **müssen** mit der Eigenschaft *resolution* des Elements *Timeline* abgestimmt werden: Beispiel: `<Timeline … resolution="120" />` **falsch**: `<Clipping visible="false" before="7" after="19"/>` (zeigt wegen Rundung fälschlicherweise 8-20 Uhr) **richtig**: `<Clipping visible="false" before="8" after="18"/>`
Header	Attribute: • width = <integer>: Breite eines Überschriftenobjekts. 0 bedeutet automatisch. Default = 0 • height = <integer>: Höhe eines Überschriftenobjekts Default = automatisch berechnet • wordWrap = <boolean>: Zeilenumbruch des Überschriftentexts (funktioniert nur im vertikalen Modus)
Navigation	Attribute: • step: Anzahl Tage, welche beim Klicken auf die Navigationsknöpfe vor- bzw. zurückgesprungen wird
Sidebar	Attribute: • dateTimeFormat = <text>: Definiert das Datumsformat in der Sidebar. Z.B. dd.mm yyy (10.04 Apr) Optional. Default = dddd, dd.mm.yyyy (Donnerstag, 10.04.2014) • width = <integer>: Definiert die Breite in Pixel der Sidebar. Z.B. 52 Optional. Default = 45
Timeline	Attribute: • resolution = <integer>: Zeit-"Auflösung" in Minuten z.B. 120 (2 Stunden) • scrollSynch = <boolean>: Setzen den aktiven Bildlauf (Ja/Nein). Optional. Default = true • shownDays = <integer>: Anzahl sichtbare Tage z.B. 5 • startingDay = <integer>: Starttag (0 = Sonntag, 1 = Montag, usw.) • timeIndicator = "none" ¦ "shortLine" ¦ "longLine": Zeigt das aktuelle Datum und Zeit als Linie an. Optional. Default = none (keine Anzeige) • timeIndicatorColor = <color>: Bestimmt die Farbe der Linie (siehe auch timeIndicator). Optional. Default = #FF0000 (Rot)

	• width = <integer>: Breite in Pixel des Überschiftenobjekts (0 = automatisch; default = 31)

14.5 Element: PlannerRule

Definiert die Regeln zur Darstellung von PlannerItems.

14.5.1 Beispiel

```
<PlannerRule>
    <CaptionColoring>Aufträge nach Status</CaptionColoring>
    <SubContents>
        [
            VIA bisP_MaintPersonToMaintOrder
        OR
            VIA bisP_MaintGroupToMaintOrder
        ]
    </SubContents>
    <Caption>$Object_Display_Kontext$</Caption>
    <Text>$bisB_Description$</Text>
    <Termin start="bisT_StartDateTime" duration="bisT_Duration"
            rebind="bisP_MaintOrderToMaintPerson"/>
    <ItemOptions />
</PlannerRule>
```

14.5.2 Attribute

Bezeichnung	Beschreibung
showIcon	

14.5.3 Elemente

Bezeichnung	Beschreibung
Caption	Überschrift des PlannerItems
CaptionColoring	Färbung für Hintergrund des Überschriftentexts
Completion	
Condition	Navigation, welche die Berechtigung zum Vergrössern und Verschieben eines Items steuert. Durch Angabe von Condition werden die Attribute FixedPos, FixedPosition, FixedTime und FixedTime gesteuert, wenn diese nicht in der Konfiguration bereits mit „`True`“ gesperrt worden sind. Bsp. Anwendung, dass nur Aufträge, welche noch nicht in Bearbeitung sind, verändert werden können: `<Condition>VALUE bisP_OrderState &lt; 1</Condition>` Erst ab Byron/BIS v5.0.0
Hint	Bestimmt den Text des Hints für ein PlannerItem. Text oder Attribut. Der Titel des Hints wird von der Caption bestimmt, wenn diese getzt ist.

	Sonst wird die Zeit und das Datum verwendet. Hints für Hintergrund-PlannerItems (backgroundItem="true") werden nicht angezeigt. Wird nur angezeigt, wenn "showHints" am Planner gesetzt ist. `Bsp: <Hint>Hallo</Hint> <Hint>$bisB_Description$</Hint>`
ItemColoring	Färbung für Hintergrund des PlannerItems
ItemOptions	Damit kann das Aussehen/das Verhalten der PlannerItems gesteuert werden. Das Element ItemOptions besitzt folgende Attribute: • allowOverlap: • backgroundItem: • fixedPosition: • fixedSize: • fixedTime: • selectColor: • selectColorTo: • selectFontColor: • setCaptionBackgroundOnSelection: • shadow: • trackVisible = <boolean>: xxx optional. Default = false Aus der TMS-Dokumentation: **FixedPos**: Boolean : When true, prevents that items are moved in the TPlanner **FixedSize**: Boolean : When true, prevents that items are sized in the TPlanner **FixedTime**: Boolean : When true, items can only be moved between positions (position being a column in vertical mode or row in horizontal mode) and the item cannot be moved so that the start time and end time would change. **FixedPosition**: Boolean : When true, items can only be moved within the same position (position being a column in vertical mode or row in horizontal mode)
SubContents	Navigation zu den PlannerItems
SubContentsInverse	
Termin	Definiert wie die Details für einen Termin ermittelt werden. Das Element Termin besitzt folgende Attribute: • duration: Legt das Attribut fest, welches die Dauer des Termins enthält oder definiert eine Expression, welche die Dauer des Termins in Tagen berechnet. Die Expression darf auch "ATTRIBUTE" enthalten. Die Dauer

	eines Termins kann nur geändert werden, wenn duration ein Attribut definiert. Beispiele: duration="bisT_Duration" duration="0.5". • rebind: Assoziation, welche für das Verschieben der PlannerItems benötigt wird (**nicht** verfügbar für PlannerMonth). • start: Legt das Attribut fest, welches den Anfang des Termins enthält oder definiert eine Expression, welche den Anfang des Termins berechnet. Ein Termin kann nur Verschoben werden, wenn start ein Attribut definiert. Beispiele: start="bisT_StartDateTime" start="INT(ATTRIBUTE(,bisT_StartDateTime')) + 8/24"
Text	Text im PlannerItem

14.6 Element: PlannerHoliday

14.6.1 Beispiel

```
<PlannerHoliday>
    START INSTANCES bisB_HolidayContainer
    VALUE bisB_ConsiderHoliday = true
    VIA Object_To_Children
    CLASS bisB_Holiday
    VALUE bisB_ConsiderHoliday = true
</PlannerHoliday>
```

14.6.2 Beschreibung

[…]

14.7 Element: Font

Definiert die Standard-Schriftart für den Header, die Sidebar, die PlannerItem-Caption und den PlannerItem-Text.

14.7.1 Beispiel

```
<Font name="Arial" size="12" color="#FF0000"
      bold="true" italic="true" underline="true" strikeOut="true" />
```

14.7.2 Attribute

Bezeichnung	Beschreibung
name	<text> Bestimmt die Schriftart z.B. Arial
size	<integer>
color	<color>
bold	<boolean>

italic	<boolean>
underline	<boolean>
strikeOut	<boolean>

14.8 Element: HeaderFont

Definiert die Schriftart für den Header.

14.8.1 Beispiel

```
<HeaderFont name="Arial" size="12" color="#FF0000"
            bold="true" italic="true" underline="true" strikeOut="true" />
```

14.8.2 Attribute

Siehe Element Font des Planners.

14.9 Element: SidebarFont

Definiert die Schriftart für die Sidebar.

14.9.1 Beispiel

```
<SidebarFont name="Arial" size="12" color="#FF0000"
             bold="true" italic="true" underline="true" strikeOut="true" />
```

14.9.2 Attribute

Siehe Element Font des Planners.

14.10 Element: CaptionFont

Definiert die Schriftart für die PlannerItem-Caption.

14.10.1 Beispiel

```
<CaptionFont name="Arial" size="12" color="#FF0000"
             bold="true" italic="true" underline="true" strikeOut="true" />
```

14.10.2 Attribute

Siehe Element Font des Planners.

14.11 Element: TextFont

Definiert die Schriftart für den PlannerItem-Text.

14.11.1 Beispiel

```
<TextFont name="Arial" size="12" color="#FF0000"
          bold="true" italic="true" underline="true" strikeOut="true" />
```

14.11.2 Attribute

Siehe Element Font des Planners.

14.12 Planner-Operationen

Siehe Gantt-Operationen

15 PlannerMonth

Mit dem Applikationselement <PlannerMonth> können Objekte in einem Monatskalender dargestellt werden.

Schematischer Aufbau erstellen (Grafik mit Week, Day, Date und Item)

15.1 Beispiel

```
<PlannerMonth ID="planner" panel="pnPlanner" onDoubleClick="Open"
              sendSelectNotification="true" receiveSelectNotification="false"
              propagateOnFocusChanged="true">
    <PlannerRule>
        <!-- ... -->
    <PlannerRule>
    <PlannerHoliday>
        <!-- ... -->
    </PlannerHoliday>
    <Propagate operation="custom" name="Date" target="calendar"/>
</PlannerMonth>
```

15.2 Beschreibung

15.2.1 XML-Attribute

Bezeichnung	Beschreibung
showHints	`showHints = <boolean>` Mit showHints="true" werden die Hints für die dargestellten Objekte (Items) angezeigt. Die Formatierung von Hints kann im Element innerhalb der <PlannerRule> definiert werden. Optional. Default = "false"

15.2.2 Ausgelöste propagates

Aktion	Operation
Datum setzen	Date

Beispiel:

```
<Propagate operation="custom" name="Date" target="plannerMonth"/>
```

Empfangene propagates:

Operation	Beschreibung
Date	Setzt das Datum im Kalender

Beispiel:

```
<Propagate operation="custom" name="Date" target="calendar"/>
```

15.2.3 Parameter für die Filternavigation

Siehe auch: Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
VAL_CurrentDate	Datum	Ausgewähltes Datum des Kalenders
VAL_FirstDate	Datum	Erstes Datum im Kalender
VAL_LastDate	Datum	Letztes Datum im Kalender

Der Parameter muss die Element-ID als Prefix beinhalten.

Beispiel:

```
<PlannerMonth ID="plannerMonth">
...
VALUE bisB_CreationDate = DATAOF 'plannerMonth.VAL_CurrentDate'
```

15.3 Element: PlannerRule

Siehe Element PlannerRule im Applikationselement Planner

15.4 Element: PlannerHoliday

Siehe Element PlannerHoliday im Applikationselement Planner

15.5 Element: PlannerMonthOptions

15.5.1 Attribut: maxItemsDisplayed

```
maxItemsDisplayed = <integer>
```

Bestimmt die Anzahl der maximal angezeigten Einträge pro Tag.

Wenn die maximale Anzahl überschritten ist, erscheinen im Monatsplanner zwei Pfeile auf der rechten Seite.

Default sind 3 Einträge pro Tag.

15.5.2 Attribute: yearBrowser / monthBrowser

```
yearBrowser = <boolean>
```

```
monthBrowser = <boolean>
```

Schaltet die Navigationspfeile (Monat) und/oder –doppelpfeile (Jahr) an oder aus.

15.5.3 Attribut: showDaysBeforeAndAfter

```
showDaysBeforeAndAfter = <boolean> (Default = true)
```

Blendet die Tage aus, die nicht zum angezeigten Monat gehören.

15.6 PlannerMonth-Operationen

Siehe Gantt-Operationen

16 Calendar

Mit dem Applikationselement <Calendar> kann ein Kalendersteuerungelement verwendet werden

16.1 Beispiel

```
<Calendar ID="calendar" panel="pnCalendar" multiCalendar="true" showWeeknumber="true">
    <CalendarRule attribute="bisT_StartDateTime" duration="bisT_Duration"
                  color="#00FF00" shape="circle">
        VIA bisP_MaintGroupToPerson VIA bisP_MaintPersonToMaintOrder
        VALUE bisT_StartDateTime >= DATAOF VAL_FirstDate
        VALUE bisT_StartDateTime &lt;= DATAOF VAL_LastDate
    </CalendarRule>
    <CalendarHoliday>
        START INSTANCES bisB_HolidayContainer
        VALUE bisB_ConsiderHoliday = true
        VIA Object_To_Children
        CLASS bisB_Holiday
        VALUE bisB_ConsiderHoliday = true
    </CalendarHoliday>
    <Propagate operation="custom" name="Date" target="planner"/>
</Calendar>
```

16.2 Beschreibung

16.2.1 XML-Attribute

- `multiCalendar = <boolean>`: Wenn true, dann wird der verfügbare Platz mit mehreren Kalenderelementen belegt
- `showWeeknumber = <boolean>`: Die Kalenderwoche ein-/ausblenden
- `pastMoreImportant = <boolean>`: Wenn `true` wird der aktuelle Monat nicht "oben links" als erster Calendar angezeigt sondern "unten rechts" als letzter, damit die Vergangenheit sichtbar ist. `Default = false`

16.2.2 Ausgelöste propagates

Aktion	Operation
Datum setzen	Date

Beispiel:

```
<Propagate operation="custom" name="Date" target="planner"/>
```

16.2.3 Empfangene propagates

Operation	Beschreibung
Date	Setzt das Datum im Kalender

Beispiel:

```
<Propagate operation="custom" name="Date" target="calendar"/>
```

16.2.4 Parameter für die Filternavigation

Siehe auch: Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
VAL_CurrentDate	Datum	Ausgewähltes Datum des Kalenders
VAL_FirstDate	Datum	Erstes Datum im Kalender
VAL_LastDate	Datum	Letztes Datum im Kalender

Der Parameter muss die Element-ID als Prefix beinhalten.

16.3 Element: CalendarRule

16.3.1 Beispiel

```
<CalendarRule attribute="bisT_StartDateTime" duration="bisT_Duration"
              color="#00FF00" shape="circle">
    VIA bisP_MaintGroupToPerson VIA bisP_MaintPersonToMaintOrder
    VALUE bisT_StartDateTime >= DATAOF VAL_FirstDate
    VALUE bisT_StartDateTime &lt;= DATAOF VAL_LastDate
</CalendarRule>
```

16.3.2 Beschreibung

- `attribute = <string>`: ...
- `duration = <string>`: ...
- `color = <color>`: ...
- `shape = "circle" | "rectangle" | "square" | "triangle"`: ...

16.4 CalendarHoliday

16.4.1 Beispiel

```
<CalendarHoliday>
    START INSTANCES bisB_HolidayContainer
    VALUE bisB_ConsiderHoliday = true
    VIA Object_To_Children
    CLASS bisB_Holiday
    VALUE bisB_ConsiderHoliday = true
</CalendarHoliday>
```

16.4.2 Beschreibung

[...]

17 Tabbed

Das Applikationselement Tabbed besteht aus mehreren Seiten (Page). Damit ist es möglich auf einer Fläche verschiedene Elemente zu haben und der Benutzer hat die Möglichkeit zwischen diesen umzuschalten. Jeder Seite kann eine Bedingung (Condition) zugeordnet werden. Wenn es eine solche gibt und diese nicht erfüllt ist (Anzahl resultierender Objekte = 0), dann wird diese Seite unsichtbar. Wenn nur noch eine Seite übrigbleibt, dann wird die Möglichkeit zum Umschalten ausgeblendet. So ist es also möglich in einem Panel die Applikationselemente abängig von der Selektion auszutauschen. Jede Seite kann ein SubLayout enthalten das es ermöglicht, mehrere Applikationselemente in einer Seite zu verteilen. Sollte das Tabbed-Element ein Propagate erhalten, wird dieses an das erste Applikationselement jeder Seite weitergeleitet.

Wichtig:
Propagates müssen auf <Tabbed> erfolgen, welches diese dann richtig verteilt. Wenn Propagates direkt an die Elemente einer <Page> erfolgen, dann wird ein korrektes Verhalten des Applikationselements in Frage gestellt. Eine derartige Konfiguration führt auch zu einer Warnung im Log, die aber unterdrückt werden kann (vgl. Propagate).
Wenn Umschaltlaschen ausgeblendet werden, dann kann dies zu Fehlern im Gr.Appl. Element führen. Wenn eine Page Grafik enthält sollte die Page immer sichtbar sein.

17.1 Attribute

Attribut	Beschreibung
forwardPropagates	`forwardPropagates = "current" | "visible" | "all"` definiert an welche Pages ein eingehendes Propagate weitergeleitet wird. `current:` Nur an das erste Applikationselement der aktuellen Seite `visible:` Nur an das erste Applikationselement aller sichtbaren Seiten (default) `all:` An das erste Applikationselement aller Seiten **Hinweis:** Propagates werden nur an die ersten Applikationselemente der Seiten propagiert. In den einzelnen Seiten müssen weitere loads/selects/customs separat verkettet werden.
hideTabSet	`hideTabSet = true` definiert ob NoTabsOnRuntime gesetzt wird wenn nur noch eine (1) Page sichtbar ist. Default ist "true".
default	`default = "<<ApplicationElement-ID>>"` definiert, welche Seite die Standardseite ist, wenn keine Condition zutrifft. Dabei wird die Seite angezeigt, die das Applikationselement mit der ID `ApplicationElement-ID` enthält.

17.2 Beispiele

Im folgenden Beispiel sind immer mindestens 2 Seiten sichtbar, wenn ein oder mehrere Doc_Container im Grid gewählt sind, dann werden 3 Seiten dargestellt. Die erste Page enthält ein SubLayout und 3 weitere Applikationselemente.

```
<Grid>
    <Propagate operation="load" target="Tabbed"/>
</Grid>
<Tabbed panel="pnExtra" default="grid3">
    <Page>
        <Caption>Seite A</Caption>
        <SubLayout orientation="horizontal">
            <Panel ID="pn1"/>
            <Layout>
                <Panel ID="pn2"/>
                <Panel ID="pn3"/>
            </Layout>
        </SubLayout>
        <Tree ID="tree" panel="pn1"/>
        <Grid ID="grid1" panel="pn2">
            <Caption>Grid1</Caption>
        </Grid>
        <Grid ID="grid2" panel="pn3">
            <Caption>Grid2</Caption>
        </Grid>
    </Page>
    <Page>
        <Caption>Seite B</Caption>
        <Condition>CLASS Doc_Container</Condition>
        <Grid ID="grid3">
            <Caption>Grid3</Caption>
        </Grid>
    </Page>
    <Page>
        <Caption>Seite C</Caption>
        <Grid ID="grid4">
            <Caption>Grid4</Caption>
        </Grid>
    </Page>
</Tabbed>
```

Im folgenden Beispiel wird immer genau eine Seite angezeigt. Die Bedingung der 2. Seite ist nie erfüllt. Wenn die erste Seite nicht gezeigt wird (kein Doc_Container), dann wird die 2. Seite (trotz nicht erfüllter Condition) angezeigt, da diese als default spezifiziert ist.

```
<Tabbed panel="pnExtra" default="F_B">
    <Page>
        <Condition>CLASS Doc_Container</Condition>
        <Form ID="F1">
            <Caption>Der Ordner $Object_Display_Kontext$</Caption>
            <FormContents name="FormA"/>
        </Form>
    </Page>
    <Page>
        <Caption>Seite B</Caption>
        <Condition>CLASS Space CLASS User</Condition>
        <Form ID="F_B">
            <FormContents name="FormB"/>
        </Form>
    </Page>
</Tabbed>
```

17.3 Empfangene propagates

Folgende Custom-Propagates sind definiert:

Operation	Beschreibung
Page	Setzt die im Attribut `code` angegebene Seite im Tabbed-Element. `code` muss einen Wert zwischen 0 und der Anzahl der Seiten -1 enthalten. Beispiel: `<Propagate operation="custom" name="Page" codeTarget="2" target="tabbed"/>`

N.B. Der Wert von *propagateName* wird case insensitive ausgewertet. (Erst ab Byron/BIS v4.11.8)

18 Search

Befindet sich noch in Entwicklung!

19 Web

Mit dem Applikationselement Web kann eine URL oder ein HTML-Inhalt angezeigt werden.

Das `<Url>`-Element bestimmt die URL, welche geladen werden soll. Falls die URL aus einem Attribut eines BisObjekts geladen werden soll, so muss das Attribut `bisAttribute` angegeben werden.

Das `<HTMLContents>`-Element bestimmt den HTML-Inhalt, welcher geladen werden soll. Falls der HTML-Inhalt aus einem Attribut eines BisObjekts geladen werden soll, so muss das Attribut `bisAttribute` angegeben werden.

Wird das Attribut `bisAttribute` angegeben, so kann mit der Contents-Navigation das Objekt bestimmt werden, für welches der Attributwert ausgelesen werden soll.

Das `<Url>`-Element und das `<HTMLContents>`-Element können beliebig kombiniert werden.

Es wird in folgender Reihenfolge versucht, den Inhalt anzuzeigen:

1. URL aus dem BisAttribut, wenn vorhanden
2. URL aus dem XML-Text, wenn vorhanden
3. HTML-Inhalt aus dem BisAttribut, wenn vorhanden
4. HTML-Inhalt aus dem XML-Text, wenn vorhanden
5. leere Seite

Custom Propagate Names dokumentieren

19.1 Beispiel

```
<Grid ID="grid" panel="pnGrid" onDoubleClick="Open" containerSupport="true">
    ...
    <Propagate operation="load" target="Web"/>
</Grid>
```

```
<Web ID="web" panel="pnWeb">
    <Contents>CLASS Person</Contents>
    <Url bisAttribute="bisB_URL"/>
    <HTMLContents bisAttribute="bisB_HTML">Keine <b>Text</b> vorhanden</HTMLContents>
</Web>
```

20 ExternalControl

Es kann ein eigenes Application Element (extern) programmiert werden. Dazu muss ein ActiveX Control programmiert werden, welches die Schnittstelle von BisXControl.BixExternalControl erfüllt. Dies ist im wesentlichen ControlOperateSingle. Hier bekommt man die Operation (load/select/custom), die URL (ObjectImage) des Objektes und die Klasse des Objektes. Die Routine ControlOperate ist dasselbe, falls mehrere Objekt propagiert werden.

N.B. Falls die Implementation eine Verbindung zur Datenbank aufbaut, dann darf dies keine inprocess ActiveX sein. Alternativen dazu sind WEB-Service oder outprocess ActiveX. Letzteres kann mit der Funktion "GetXBisDataBaseOutprocess" der Klasse "ByGlobalFunctions" in der Bibliothek "BISxUtilities" erreicht werden.

Für die Erzeugung eines ExternalControls bitte mit Byron in Verbindung setzen!
Siehe interne Dokumentation "G:\Projekte\bis\source\ActiveX_Control\Implementierung erzeugen.txt"

20.1 Beispiel

```
<ExternalControl ID="x1"  panel="main" name="BisXControl.BixExternalControl">
    <Data>$demo_SpaceID$|$bisB_ObjectImage$</Data>
</ExternalControl>
```

21 ExternalBrowser (ab v5.5.10)

Mittels ExternalBrowser kann ein Chromium basierter Web-Browser als Applikationslement verwendet werden.

21.1 Beispiel

```
<ExternalBrowser ID="browser" panel="browser" headerVisible="true"
                 devTools="true" showUrl="true" initialUrl="about:blank"
                 localDownloadForbidden="false"
                 urlAttribute="Doc_Path">
    <Caption>Chromium-Browser</Caption>
</ExternalBrowser>
```

21.2 Attribute

Attribut	Beschreibung
showUrl	`showUrl = "true" | "false"` definiert, ob die aktuell dargestellte URL (Readonly Adresszeile) angezeigt wird. Vorgabe: `"false"`.
initialUrl	`initialUrl = "<url>"` definiert die initial angezeigte URL. Vorgabe: `"about:blank"`.
devTools	`devTools = "true" | "false"` definiert, ob die Dev-Tools als separates Fenster angezeigt werden können. Es wird in der Adresszeile ein entsprechender Button zum Öffnen der Dev-Tools angezeigt. Vorgabe: `"false"`.
executeName	Pfad des externen Programms, welches im ExternalWindow gerendert werden soll. Parameter wie *%BIS_ROOT%* werden immer ausgewertet. Vorgabe: `"%BIS_ROOT%SimpleCefSharpHost\SimpleCefSharpHost.exe"`
urlAttribute	`urlAttribute = "<attrib>"` definiert das Byron/BIS-Attribut, welches vom geladenen Objekt als anzuzeigende URL verwendet wird. Vorgabe: `"Doc_Path"`.
localDownloadForbidden	`localDownloadForbidden = "true" | "false"` definiert, ob das Herunterladen von nicht im Browser darstellbarem Inhalt mittels "Speichern unter" verboten ist. Vorgabe: `"false"`.

21.3 Empfangene Propagates

Bezeichnung	Beschreibung
load	Vom ersten Objekt der empfangenen Objektmenge wird das in `urlAttribute` konfigurierte Attribut gelesen und als aktuelle URL übernommen. Variablen und globale Parameter werden zuvor ausgewertet.

	Existiert kein Objekt, kein Attribut oder ist der resultierende Wert leer, wird wieder `initialUrl` angezeigt. Wenn die URL gegenüber dem letzen Laden nicht geändert hat, wird nicht neu geladen.
Select	Die selektierten Objekte werden als Select-Command (Id=3) an das externe Programm gesendet. Die übergebene Operation ist leer.
Custom	Die selektierten Objekte werden als Select-Command (Id=3) an das externe Programm gesendet. Die übergebene Operation entspricht dem Namen der CustomOperation.

22 ExternalWindow (ab v5.5)

Es kann ein eigenes Application Element (extern) programmiert werden. Die Kommunikation erfolgt über Propagates. Diese werden dann über eine Pipe weitergegeben und über Windows-Messages signalisiert.

Startvorgang

Das externe Programm wird in BIS gestartet, gemäss dem Attribut *executeName*. Es erhält beim Aufruf zwei Parameter:

1. Das Handle des Windows, auf welchem es zeichnen soll im Parameter /wnd=*windowId*
2. Der Name der NamedPipe, aus welcher alle weiteren Informationen bezogen werden. /pipe=*namedPipe*. Der Pipename wird ohne '\\.\pipe\' angegeben. Bsp: G:\exe\ExtWin.exe /wnd=6351 /pipe=byrbis52648

Beim Start wird falls vorhanden der Wert von Element <Initialize> übermittelt.

N.B.
Diese Propagates können nicht nur zwischen den Applikationselementen, sondern auch durch die Operation opPropagate oder im Formularscripting ausgelöst werden.

Die Umsetzung der Objekte in ihre textuelle Form word in ObjectToText definiert. Für verschiedene Klassen können verschiedene Attribute verwendet werden. Der default ist in `attribute` am ExternalWindow definiert.

22.1 Attribute

Zusätzliche Attribute am Element **<ExternalWindow>**

Bezeichnung	Beschreibung
executeName	Pfad des externen Programms, welches im ExternalWindow gerendert werden soll. Parameter wie %*BIS_ROOT*% werden immer ausgewertet. Vorgabe: nicht gesetzt

22.2 Element Initialize

Attribute am Element `<Initialize>`. Der Inhalt (die enthaltenen XML-Elemente) von Initialize wird an die exe-Anwendung gesendet

Bezeichnung	Beschreibung
parameterDelimiter	Wenn dieses Attribut gesetzt ist, wird der Inhalt von <Initialize> vor dem Versenden noch evaluiert, mit diesem Zeichen als Variabalenbegrenzer. Da die Werte immer rekursiv ausgewertet werden, soll hier ein Zeichen definiert werden, das nicht in den Werten vorkommt.
asXML	Boolean ob der Inhalt des Initialize tags als XML übergeben wird. Vorgabe: "`false`"

22.3 Element PrePropagate

Mit dem Element `<PrePropagate>` wird der Select-Objektaustausch zwischen dem externen Programm und ByronBIS geregelt: Die eingehenden Objekte werden über das definierte `attribute` zu einem Text umgewandelt, der an das externe Programm gesendet wird. Vom externen Programm werden Objekte empfangen und mit der FilterNavgation in Objekte umgewandelt. Diese Obejekte werden dann an die Propagates weitergesendet.
Beispiel:

```
<PrePropagate attribute="bisB_BIM_ID">
     START VALUE bisB_BIM_ID = '=id'
     COUNT ( [  VIA Object_To_Parent VIA bisB_ProtokollToDoc
                AND RECALL 'OBJ_Doc' ] ) = 1
     VIA bisB_RecordToObject
</PrePropagate>
```

N.B.

Wenn man von einer Eindeutigkeit des Attributes *bisB_BIM_ID* ausgehen könnte, dann wäre das COUNT-Statement nicht notwendig.

22.4 Element BIMViews

Mit dem Element `<BIMViews>` kann wie bei den Grafikansichten eine Auswahl an BIM-Ansichten ermittelt werden. Als Vorgabe werden alle existierenden BIM-Ansichten angeboten.

22.5 Empfangene Propagates

Bezeichnung	Beschreibung
load	Bei einem Doc_File wird der Dateipfad weitergesendet. Bei allen übrigen Objekten wird die durch *bisB_BIM_ID* ermittelte ID zusammen mit der *Caption* und der *Färbung* des Objekts übermittelt.
Select	Die selektierten Objekte werden als Select-Command (Id=3) an das externe Programm gesendet. Die übergebene Operation ist leer.
Custom	Die selektierten Objekte werden als Select-Command (Id=3) an das externe Programm gesendet. Die übergebene Operation entspricht dem Namen der CustomOperation. Mit `propagateName="bisObjects"` kann eine Menge an Objekten gemäss <Transform> übergeben werden, welche anschliessend gefliterd angezeigt werden. Mit `propagateName="allObjects"` werden wieder alle Objekte des IFCs geladen.

22.6 Beispiel Forge (BIMViewerHost.exe)

22.6.1 Beispielkonfiguration

```
<!-- Kopfformular für Xtool -->
<Form ID="Xhead"
      panel="pnExtHead"
      ignoreMultiInputObjects="false"
      headerVisible="false"
      optional="false">
   <FormContents name="IFCselection"/>
   <Propagate operation="load"
              target="Xtool"
              name="ifcCombo"/>
   <!-- workaraound -->
   <Propagate operation="show"
              target="tree"
              name="selBtn">
      <Transform>
         [
            CLASS Space VIA Object_To_Parent
         OR
            CLASS Component VIA Room_Of_Component
         ]
      </Transform>
   </Propagate>
   <Propagate operation="select"
              target="grid"
              name="selBtn"
              oneStep="true"/>
   <Propagate operation="custom"
              target="Xtool"
              name="colorChanged"
              oneStep="true"/>
</Form>

<!-- X-Tool -->
<ExternalWindow ID="ExternalWindow"
                panel="pnFormForgeViewer"
                headerVisible="false"
                executeName="%BIS_ROOT%\BIMViewer\BIMViewerHost.exe">
   <Initialize asXML="true"
               parameterDelimiter="%">
      <Login>%BIM_Forge_ClientID%</Login>
      <pw>%BIM_Forge_ClientSecret%</pw>
      <WebSiteUrl>%BIM_Forge_WebSiteUrl%</WebSiteUrl>
   </Initialize>
   <Caption>angezeigt: $Object_Display_Kontext$</Caption>
   <Propagate operation="load"
              target="form"/>
   <Propagate operation="select"
              target="grid"/>
   <Propagate operation="select"
```

```
            target="Xhead"/>
   <PrePropagate attribute="bisB_BIM_ID">
      START VALUE bisB_BIM_ID = '=id'
      COUNT ( [VIA Object_To_Parent VIA bisB_ProtokollToDoc AND RECALL 'OBJ_Doc'] ) = 1
      VIA bisB_RecordToObject
   </PrePropagate>
</ExternalWindow>
```

22.6.2 Hinweise

Aus dem Formular heraus werden die Objekte mit dem Parameter wie folgt übergeben:

```
FormInformation.PropagateData('ifcCombo', 0, null, IfcDocObject.DbObject);
```

- 0 = laden
- 1 = vergleichen

Die per `Propagate load` erhaltenen Objekte werden übergeben, wenn es sich um eine Ableitung von `Doc_File` handelt. Oder es sich um Geometrie-Objekte (Vektorgrafik, Position) handelt. Es können mehrere IFC-Dokumente sein, welche sich laden oder vergleichen lassen.

22.7 Beispiel IfcViewer (VectorDraw)

22.7.1 Beispielkonfiguration

```
    <!-- X-Tool -->
    <ExternalWindow ID="Xtool" panel="pnExt" headerVisible="true"
                executeName="%BIS_ROOT%\VectorDraw\VDIFCViewerHost.exe">
      <Initialize asXML="true" parameterDelimiter="%">
        <Login>%BIM_Forge_ClientID%</Login>
        <pw>%BIM_Forge_ClientSecret%</pw>
        <WebSiteUrl>%BIM_Forge_WebSiteUrl%</WebSiteUrl>
      </Initialize>
      <Caption>angezeigt: $bisB_DocToProtokoll%bisB_ProtokollToObject%Object_Dis-
play_Kontext$  $bisB_DocToProtokoll%bisB_CreationDate$  $Object_Display_Kon-
text$</Caption>
      <Propagate operation="load" target="form"/>
      <Propagate operation="select" target="grid"/>
      <Propagate operation="select" target="Xhead"/>
      <PrePropagate attribute="bisB_BimIfc_ID">
        SAVEAS convertedId "=CONVERTIFCGUID(id)"
        START VALUE bisB_BIM_ID = '=convertedId'
        COUNT ([VIA Object_To_Parent VIA bisB_ProtokollToDoc AND RECALL 'OBJ_Doc'])
= 1
        VIA bisB_RecordToObject
      </PrePropagate>
      <LoadSpace height="bisB_Height" Z="bisB_Zbase"  />
      <Toolbar caption="Grafik">
        <ToolbarButton><Execute name="showOnlyDbObjs" /></ToolbarButton>
        <ToolbarButton><Execute name="showAllIfcObjs" /></ToolbarButton>
      </Toolbar>
    </ExternalWindow>


    <Operations>
      <!-- BIM -->
```

```
      <Operation ID="showInBIM" opCode="opPropagate" toElement="Xtool" propagate-
Type="custom" propagateName="show"
                caption="in Grafik zeigen"  icon="opShow">
          <Source>RECALL 'grid.OBJ_Selection' </Source>
          <Condition>
              VIA bisB_ObjectToRecord
              COUNT ([VIA Object_To_Parent VIA bisB_ProtokollToDoc AND RECALL
'OBJ_Doc']) = 1
          </Condition>
          <Transform>
              VIA bisB_ObjectToRecord
              COUNT ([VIA Object_To_Parent VIA bisB_ProtokollToDoc AND RECALL
'OBJ_Doc']) = 1
          </Transform>
      </Operation>
      <Operation ID="showOnlyDbObjs" opCode="opPropagate" toElement="Xtool" propa-
gateType="custom" propagateName="bisObjects"
                caption="nur Datenbankobjekte zeigen"  icon="icVonDb" _initial-
Down="true">
          <Source>
            RECALL 'OBJ_Doc'                -- IFC-Dokument
            VIA "bisB_DocToProtokoll"       -- Import-Protokoll
            --LOG objectSource DEEP 2
          </Source>
          <Transform>
            VIA "Object_To_Children" -- Import-Records
            COUNT (VIA bisB_RecordToObject)
            --LOG objectsRecords DEEP 2
          </Transform>
      </Operation>
      <Operation ID="showAllIfcObjs" opCode="opPropagate" toElement="Xtool" propa-
gateType="custom" propagateName="allObjects"
                caption="Alle BIM-Objekte zeigen"  icon="icVonIfc" >
      </Operation>
    </Operations>
```

22.7.2 Hinweise

Siehe auch «22.6.2 Hinweise»

23 ExternalVectorDraw (ab V 5.5)

Mit dem Application Element `<ExternalVectorDraw>` können Objekte grafisch dargestellt werden. Vector-Draw wird hierbei als externes Programm verwendet.

Die Darstellung erfolgt aufgrund der Attribute:

- Symbol an einer Position,
  zwingende Attribute: Position, bisG_SymbolName (und Symbol muss existieren).
  weitere Attribute: Z_Drehwinkel, bisG_Scale, bisG_YStretch
- Flächen,
  zwingende Attribute: bisB_AreaGeometrie

Verbindungslinien werden zurzeit nicht unterstützt.

Zusätzliche Attribute am Element **< ExternalVectorDraw>**

Bezeichnung	Beschreibung
executeName	Pfad des externen Programms. default: `%BIS_ROOT%\VectorDraw\VDrawHost.exe`
backgroundIsEqualSpace	Boolean default false false: Die Räume sind im globalen Koordinatensystem, der Plan in einem lokalen Architektur Koordinatensystem. true: Der Plan und die Räume befinden sich im gleichen Koordinatensystem.
priorityAttribute	Legt das Attribut am Objekt fest, welches die Priorität (Z-Level) für die Darstellung resp. Reihenfolge der Objekte enthält. Wenn kein Wert am Objekt definiert ist, gilt der Default: • Für Flächenobjekte: 200 • Für Positionsobjekte: 400 • Beschriftungen: 800 • Hintergrund: -1 oder 10000 Für die Visualisierung von Selektionen, Löchern (Säulen, etc.) oder Unterräumen finden weiter folgende Prioritäten-Additionen statt (Priorität Additionen): • Selektion: +1: • Loch +2: • Unterraum +3: *Bsp. für Auswirkungen in der Praxis:* *Wird z.B. mit Unterräumen gearbeitet, also Räume welche andere Räume auf dem Plan überlagern, so muss dafür gesorgt werden, dass deren Priorität um -2 tiefer sind als der Default (also 198 statt 200).* *Dadurch ist gewährleistet, dass der Unterraum noch sichtbar uns auswählbar ist (198 + 3) und dennoch der Selektion des Hauptraums durch die Selektion überdeckt wird (200 + 1). Im Selektionsfall haben beide die Priorität von 201, jedoch greift die Selektion stärker.*

23.1 Element: Condition

Siehe Element: Condition

23.2 Element: Contents

Siehe Element: Contents

23.3 Element: GraphicViews

Siehe Element GraphicViews

23.4 Element: Pick (Alternative ab v5.7.3)

Das Element `<Pick>` wie im Element `<Graphic>` steht **NICHT** zur Verfügung. Stattdessen können Operationen mit dem `opCode="evdPickable"` erstellt werden, ohne dass die Operation durch einen Toolbarbutton oder ein Menü verwendet wird. Siehe auch Operation: evdPickable

23.5 Grafik Operationen

Die Operationen werden durch die Attribute `initialDown` und `parameter` charakterisiert.:

Wenn das Attribut **`initialDown`** angegeben ist, dann handelt es sich um einen Schaltbefehl (Modus). Der Wert für initialDown ist die Vorgabe für den Wert der Einstellung.

Wenn das Attribut **`parameter`** "true" gesetzt ist, dann verlangt der Befehl eine zusätzliche Eingabe. Dies Befehle können dann mit Cancel wieder abgebrochen werden.

Einem Befehl kann mit **`value`** auch noch ein Wert übergeben werden. (Beispiel `evdPickable`). Wenn ein Element `<Definition>` angegeben ist, wird der Text dieses Elements als value übernommen. Vgl. evdPrint

```
<Operation ID="G_RotateMode" opCode="evdRotateMode" forExternalVectorDraw="XvectorDraw"
           icon="grRotateMode" caption="@grRotateMode_Caption@"   initialDown="true"/>
<Operation ID="G_ZoomDetail" opCode="evdZoomDetail" forExternalVectorDraw="XvectorDraw"
           icon="grZoomDetail" caption="@grZoomDetail_Caption@" parameter="true"/>
<Operation ID="G_ZoomTotal" opCode="evdZoomTotal" forExternalVectorDraw="XvectorDraw"
           icon="grZoomTotal" caption="@grZoomTotal_Caption@"/>
<Operation ID="G_Cancel" opCode="evdCancel" forExternalVectorDraw="XvectorDraw"
           icon="opCancel" caption="@grCancel_Caption@"/>
<Operation ID="G_AreaPickable" opCode="evdPickable" forExternalVectorDraw="XvectorDraw"
           icon="grAreaPickable" caption="@grAreaPickable_Caption@"
           initialDown="true" value="200"/>
<Operation ID="G_SymbolPickable" opCode="evdPickable" forExternalVectorDraw="XvectorDraw"
           icon="grSymbolPickable" caption="@grSymbolPickable_Caption@"
           initialDown="true" value="400"/>
<Operation ID="G_ShowLegend" opCode="evdShowLegend" forExternalVectorDraw="XvectorDraw"
           icon="opObjectList" caption="@grShowLegend_Caption@" initialDown="true"/>
<Operation ID="G_SaveAs" opCode="evdSaveAs" forExternalVectorDraw="XvectorDraw"
           icon="opSaveConfigurations" caption="Speichern unter"/>
<Operation ID="G_Move" opCode="evdMove" forExternalVectorDraw="XvectorDraw"
           icon="opObjectList" caption="verschieben" parameter="true"/>
<Operation ID="G_Print" opCode="evdPrint" forExternalVectorDraw="XvectorDraw"
           icon="opPrint" caption="drucken">
  <Definition>
       {
         "A4H" : { "NumOfXTabs" : 1,
                   "NumOfYTabs" : 8,
```

```
                "VectorDrawRect" : { "StartX" : 0,
                                     "StartY" : 4,
                                     "Width" : 1,
                                     "Height" : 4
                                   },
                "Objects" : [
                             { "StartX" : 0,
                               "StartY" : 0,
                               "Width" : 1,
                               "Height" : 1,
                               "Text": "Description"
                             }
                           ]
              },
      "A4Q" : { "NumOfXTabs" : 8,
                "NumOfYTabs" : 8,
                "VectorDrawRect" : { "StartX" : 4,
                                     "StartY" : 0,
                                     "Width" : 4,
                                     "Height" : 8
                                   },
                "Objects" : [
                             { "StartX" : 0,
                               "StartY" : 7,
                               "Width" : 4,
                               "Height" : 1,
                               "Text": "Description"
                             }
                           ]
              }
    }
    </Definition>
</Operation>
```

23.5.1 Operation: evdPrint

Siehe unter Kapitel Operationen bei `evdPrint`

23.5.2 Operation: evdRotate

Diese Operation ermöglicht die freie Rotation des Objekts.

23.5.3 Operation: evdRotateMode

Zustandsbutton. Wenn «down», dann wird der Plan in den Architektur-Space gedreht, wenn nicht «down», dann ist der Plan in Weltkoordinaten rotiert. `initialDown=«true»` ist hier üblich.

Der Wert von «`initialDown`» wird auch dann verwendet, wenn kein Toolbarbutton mit dieser Operation existiert.

23.5.4 Operation: evdZoomDetail

Diese Funktion zoomt in ein User-definierten Rechteck hinein. Wenn die Funktion durch Anklicken aktiviert wurde, kann der Benutzer mit zwei weiteren Klicks in den Plan das Rechteck aufspannen, in welches hereingezoomt werden soll. Die Operation kann mittels `evdCancel` abgebrochen werden.

23.5.5 Operation: evdZoomTotal

Diese Operation löst ein Zoom aus, der alle gezeichneten Elemente darstellt.

23.5.6 Operation: evdCancel

Diese Operation bricht andere Operationen ab, welche mehrere Klicks benötigen.
Zum Bsp. `evdZoomDetail` oder `evdMove`.

23.5.7 Operation: evdPickable

Diese Operation ist eine Zustandsabbildung. Wenn sie «down» ist, sind die Elemente, welche in «value» übergeben werden, auswählbar, wenn nicht, dann sind sie nicht auswählbar.

- Flächen werden mit dem value «200» übergeben
- Symbole werden mit dem value «400» übergeben
- Beschriftungen / Texte werden mit dem Wert «800» übergeben

Der Wert von «`initialDown`» wird auch dann verwendet, wenn kein Toolbarbutton mit dieser Operation existiert.

23.5.8 Operation: evdShowLegend

Diese Operation ist eine Zustandsabbildung. Wenn sie «down» ist, wird die Legende links unten im Plan dargestell, wenn nicht, dann wird sie nicht dargestellt.

Der Wert von «`initialDown`» wird auch dann verwendet, wenn kein Toolbarbutton mit dieser Operation existiert.

23.5.9 Operation: evdSaveAs

Diese Operation speichert den Plan in verschiedenen auswählbaren Formaten.

23.5.10 Operation: evdMove

Diese Operation verschiebt Symbole oder Beschriftungen auf dem Plan.
Die Operation kann nur verwendet werden, wenn ein Symbol oder eine Beschriftung angewählt ist.
Mit dem ersten Klick wird der Ursprungspunkt, mit dem zweiten Klick wird der Zielpunkt gewählt.

Beide Punkte können mit Hilfe der Fang-Optionen beeinflusst werden. Mehr dazu unter «23.5.16 Operation: evdOpenOsnapDialog».

Folgendes ist zurzeit nicht implementiert: Abbrechen mit `evdCancel` und die Verwendung von `parameter` und `initialDown`.

23.5.11 Operation: evdExportPdf

Diese Operation exportiert den Plan als PDF.

Empfohlen wird stattdessen der Weg über `evdPrint` und da die PDF-Export-Funktion zu verwenden.

23.5.12 Operation: evdPrintNative

Diese Operation ruft das native Druckfenster auf, welches von VectorDraw zur Verfügung gestellt wird.

23.5.13 Operation: evdDropPosition

Diese Operation **unterstützt nur Positionskomponenten**. Sie setzt das Attribut *Position* auf die Koordinate des Drop Punktes und verändert die Assoziation *Room_Of_Component* indem aus der Menge der in der Grafik angezeigten Spaces derjenige ausgewählt wird, in welchem die Position liegt. Wird kein Space gefunden, dann wird die Assoziation *Room_Of_Component* gelöscht.

Beispiel:

```
<Operation ID="G_Position_EV" opCode="evdDropPosition"
           forExternalVectorDraw="XvectorDraw"
           icon="Location"
           caption="Positionieren">
  <Source>CLASS Position_Component</Source>
  <Target>CLASS Room</Target>
</Operation>
<DragDrop keyState="Shift"><Execute name="G_Position_EV"/></DragDrop>
```

23.5.14 Operation: evdUndo

23.5.15 Operation: evdRedo

23.5.16 Operation: evdOpenOsnapDialog

Öffnet eine Dialogbox zum Auswählen der Fang-Optionen (Snap-Optionen). Diese gelten als Vorgabe für alle kommenden Verschiebe-Operationen, z.B. von Komponenten oder Beschriftungen.

Die Dialogbox mit den Fang-Optionen lässt sich auch mittels CTRL + Rechtsklick öffnen. Passiert dies während der Verschieben-Operation, so gilt dies nur für die Wahl des nächsten Punktes.

23.5.17 Operation: evdMoveMap

Pan-Funktion zum Verschieben der Karte mittels linker Maustaste.

Bsp.:

```
<Operation ID="G_MoveMap_EV" opCode="evdMoveMap" forExternalVectorDraw="XvectorDraw"
     icon="evdPan" caption="@grMoveMap_Caption@"
     initialDown="false"/>
```

23.6 Beispiel

```
<ExternalVectorDraw ID="XvectorDraw" panel="pnExt" headerVisible="true">
  <Caption>angezeigt: $Object_Display_Kontext$</Caption>
  <Condition>CLASS Floor</Condition>
  <Contents>
    CLASS Floor
    VIA Object_To_Children
    CLASS Room
  </Contents>
  <GraphicViews sizeThread="5" comboWidth="200" userinterface="true">
    <Use>
      START INSTANCES bisG_GraphicView
      VALUE bisG_GraphicViewTarget = 1 -- 1- VectorDrav / 0 - Condor
      VALUE bisG_GraphicViewGroup "Raumbuch" SORT (+Object_Display_Kontext)
    </Use>
  </GraphicViews>
  <Propagate operation="select" target="grid"/>
  <Toolbar caption="Grafik">
    <ToolbarButton><Execute name="G_ZoomDetail_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_ZoomTotal_EV"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="G_Cancel_EV"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="G_AreaPickable_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_SymbolPickable_EV"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="G_RotateMode_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_ShowLegend_EV"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="G_Move_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_Print_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_PrintNative_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_ExportPdf_EV"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="G_Undo_EV"/></ToolbarButton>
    <ToolbarButton><Execute name="G_Redo_EV"/></ToolbarButton>
  </Toolbar>
</ExternalVectorDraw>
```

24 NavBar

<Beschreibung>

24.1 Beispiel

<Beispiel>

25 ShortcutPanel

Mit dem Applikationselement <ShortcutPanel> können Objekte in einer Verknüpfungsliste dargestellt werden. Die Verknüpfungen werden gespeichert, damit bei einem Neustart diese wieder zur Verfügung stehen. Verknüpfungen können nur interaktiv durch die unten aufgeführten Shortcut-Opertationen erstellt, verschoben und entfernt werden.

In den Explorern wird das Element meistens im Menü "Ansicht" → "Verknüpfungsleiste" ein- bzw. ausgeblendet werden.

25.1 Beispiel

```
<ShortcutPanel ID="shortcutPanel" panel="pnShortcutPanel" onDoubleClick="Open"
               optional="true">
    <Propagate operation="load" target="tree">
        <Condition>CLASS bisB_PersonContainer</Condition>
    </Propagate>
</ShortcutPanel>
```

25.2 Operation: opShortcutCreate

Erzeugt eine Verknüpfung in der Verknüpfungsliste zu einem Objekt.

Die Operation kann im Drag&Drop verwendet werden.

Beispiel

```
<Operation ID="CreateShortcut" opCode="opShortcutCreate"
           onElement="shortcutPanel"/>
```

25.3 Operation: opShortcutMove

Verschiebt eine Verknüpfung in der Verknüpfungsliste.

Die Operation kann im Drag&Drop verwendet werden.

Beispiel

```
<Operation ID="MoveShortcut" opCode="opShortcutMove"
           onElement="shortcutPanel"/>
```

25.4 Operation: opShortcutRemove

Entfernt eine Verknüpfung in der Verknüpfungsliste zu einem Objekt.

Die Operation kann in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="RemoveShortcut" opCode="opShortcutRemove"
           onElement="shortcutPanel"/>
```

26 Monitor

Damit werden GlobalNotify überwacht. Wenn in der Methodensprache

```
GlobalNotify(self, 1, "");
```

aufgerufen wird, dann kann dies im Monitor abgefangen werden.

Anstelle von self kann natürlich auch ein anderes Objekt übergeben werden. Die anderen 2 Parameter sollte man so belassen. Wenn die obige Zeile in die Setter () des Attributes bisP_MessageState an der Klasse bisP_Message eingefügt wird, dann wird man über alle Zustandsänderungen der Meldungen informiert.

26.1 Beispiel

```
<Monitor ID="MonitorID" panel="MonitorPanel" optional="true"
         minimizable="true" alertIcon="bisB_WarningSign" onDoubleClick="Open"
         minimized="true" visible="true"
         onAlert="showDetailed">
    <Display>
        <b>$UnreadNotifications$</b> ungelesen ->
        $DateLastNotification$ - $TimeLastNotification$
    </Display>
    <ObjectDisplay caption="URL">
        $Object_Display_Kontext$
    </ObjectDisplay>
    <!-- Alle Aufträge von Christian Haller werden als Startmenge angezeigt -->
    <StartObjects>
        START INSTANCES bisP_MaintOrder
        COUNT(VIA bisP_MaintOrderToMaintPerson
        VALUE bisB_Email = 'christian.haller@byron.ch') > 0
    </StartObjects>
    <!-- Nur Aufträge werden angezeigt (egal ob Startmenge oder Selection) -->
    <Contents>
        CLASS bisP_MaintOrder
    </Contents>
    <!-- Propagates -->
    <Propagate operation="load" target="grid" name="NewNotification"/>
    <Propagate operation="load" target="grid">
        <Transform>
            VIA "bisP_MaintOrderToMessage"
        </Transform>
    </Propagate>
</Monitor>
```

26.2 Attribute

Bezeichnung	Beschreibung
actualizeWhenInactive	Default abweichend zu allgemeinem Application Element. Default = "true"

visible	`visible = <boolean>` definiert ob der Monitor unsichtbar gestartet werden soll. Default = "true"
alertIcon	`alertIcon = <string>` Das Icon das im Headerbereich angezeigt wird wenn neue Notifications vorhanden sind. Default = "" Ab Byron/BIS v4.11.8: dem Icon können mit ‚+' Overlays hinzugefügt werden. Z.B. `icon = "bisP_Order+state_new"`
onAlert	`onAlert = "showDetailed" | "show" | "doNothing"` definiert das Verhalten wenn eine neue Notification eintrifft. `showDetailed:` Monitor wird visible und ist maximiert `show:` Monitor wird visible `doNothing:` nichts passiert (default)
maxItemsDisplayed	`maxItemsDisplayed = <integer>` definiert die maximale Anzahl von angezeigten Notifications in der Liste. StartObjects und das selektierte Item bleiben immer bestehen. Die Liste wird von "hinten" her aufgeräumt bis die letzte ungelesene Notification erscheint. Von da an wird nichts mehr gelöscht!

26.3 Elemente

Bezeichnung	Beschreibung

Display	Der Inhalt dieses Elements wird dazu verwendet, die Caption des Monitors zu setzen. Dies kann in HTML geschehen um z.B. eine fette Font (<b>) einzufügen. Zuerst werden die folgenden Variablen im String ersetzt und dann ein Evaluate der Dollar-Notation für das letzte eingetroffene Objekt ausgeführt.
	Variablen:
	`$TotalNotifications$` = Anzahl Einträge im Grid
	`$UnreadNotifications$` = Anzahl fetter Einträge im Grid
	`$DateLastNotification$` = Datum zuletzt eingetroffener Notification
	`$TimeLastNotification$` = Zeit zuletzt eingetroffener Notification
	Beispiel:
	`<Display>`
	`$Object_Display_Kontext$ - <b>$UnreadNotifications$</b> ungelesen`
	`</Display>`

ObjectDisplay	Darstellungsformel für das im Grid dargestellte Objekt. Optional kann über das Attribut "caption" der Spaltentitel gesetzt werden. Default für das caption-Attribut = "Objekt".

	Beispiel: `<ObjectDisplay caption="Bezeichnung">$Name$</ObjectDisplay>`
Contents	FilterNavigation die definiert, welche Objekte im Grid angezeigt werden. Objekte aus der <StartObjects>-Filternavigation und Objekte die mit über eine globale ChangeNotification an den Monitor weitergeleitet werden müssen erst durch diesen Filter bevor sie im Grid angezeigt werden.
StartObjects	FilterNavigation die eine Startmenge von Objekten im Grid vorgibt. Auf dieser Startmenge wird die Filternavigation vom <Contents>-Element auch ausgeführt. **ab v4.10.6** kann in dieser FilterNavigation auch schon korrekt vorsortiert werden. Die Startobjekte werden dann in **umgekehrter** Reihenfolge dem Monitor hinzugefügt (letztes Objekt aus der Navigation = zuletzt eingetroffen = zuoberst in Monitor).

26.4 Propagates

Bezeichnung	Beschreibung
	Beim Selektieren von Objekten im Grid wird propagiert. `<Propagate operation="load" target="grid">`
NewNotification	Sobald eine neue Globale ChangeNotification empfangen wurde, und es einige Objekte durch den Contents-Filter geschafft haben werden diese Objekte unter dem Namen "NewNotification" weiterpropagiert. `<Propagate operation="load" target="grid" name="NewNotification"/>`

27 Chart (ab Byron/BIS v4.5.4)

Das Chart-Element enthält die Komponente "TBisDynChartII" aus dem Formulardesigner.

27.1 Elemente

Bezeichnung	Beschreibung
Contents	Diese FilterNavigation definiert die Objekte, welche als Eingangsmenge für die BisSerie im Chart verwendet werden.
Criterion	Definiert die Gruppierungskriterien für die Objekte aus dem Contents-Element. (Optional)
BisSerie	Definiert die BisSerie für den Chart
BisFunctionSerie	Definiert die BisFunctionSerie für den Chart
BottomAxis	Definiert die untere Achse
LeftAxis	Definiert die linke Achse
RightAxis	Definiert die recghte Achse
TopAxis	Definiert die obere Achse
Legend	Definiert die Legende des Charts

27.2 Chart-Operationen

27.2.1 opChartPrintTo

Mit dieser Operation kann der Chart gedruckt, in die Zwischenablage kopiert oder als Datei gespeichert werden. Folgende Ausgabeformate (outputFormat) werden z.Z. unterstützt:

10	Print Landscape
11	Print Portrait
20	Copy to Clipboard (Bitmap)
21	Copy to Clipboard (Metafile)
22	Copy to Clipboard (Enhanced Metafile)
30	Save to Clipboard (Bitmap)
31	Save to Clipboard (Metafile)
32	Save to Clipboard (Enhanced Metafile)

Beispiele

```
<Operation ID="ChartPrintToPrintLandscape" opCode="opChartPrintTo"
           outputFormat="10" caption="Print Landscape"/>

<Operation ID="ChartPrintToPrintPortrait" opCode="opChartPrintTo"
           outputFormat="11" caption="Print Portrait"/>

<Operation ID="ChartPrintToCopyToClipboardBitmap" opCode="opChartPrintTo"
           outputFormat="20" caption="Copy to Clipboard (Bitmap)"/>

<Operation ID="ChartPrintToCopyToClipboardMetafile" opCode="opChartPrintTo"
           outputFormat="21" caption="Copy to Clipboard (Metafile)"/>

<Operation ID="ChartPrintToCopyToClipboarddMetafileEnh" opCode="opChartPrintTo"
           outputFormat="22" caption="Copy to Clipboard (Enhanced Metafile)"/>

<Operation ID="ChartPrintToSaveToBitmapFile" opCode="opChartPrintTo"
           outputFormat="30" caption="Save to Clipboard (Bitmap)"/>

<Operation ID="ChartPrintToSaveToMetafile" opCode="opChartPrintTo"
           outputFormat="31" caption="Save to Clipboard (Metafile)"/>

<Operation ID="ChartPrintToSaveToMetafileEnh" opCode="opChartPrintTo"
           outputFormat="32" caption="Save to Clipboard (Enhanced Metafile)"/>
```

27.3 Beispiel

```
<Chart ID="chart" panel="pnChart" axisVisible="true"
       bisChartType="bstLineSeries" color="$000000" marginBottom="4"
       marginLeft="3" marginRight="3" marginTop="4" view3D="true"
       view3DWalls="true">
    <Contents>
        CLASS Person
        VIA Room_Of_Component CLASS Room
        VIA Object_To_Parent  CLASS Floor
    </Contents>
    <!--Criterion attribute="" format="">
        <Navigation></Navigation>
    </Criterion>
    <Criterion attribute="" format="">
        <Navigation></Navigation>
    </Criterion-->

    <!-- weitere Criteria -->

    <BisSerie bisChartType="bst_byChart" color="$000000" colorAutomatic="true"
              drawBetweenPoints="false" markStyle="smsValue" markVisible="false"
              name="BisSerie1" pointerStyle="psRectangle" pointerVisible="false"
              showInLegend="false" sortXOrder="loNone" title=""
              vertAxis="aLeftAxis" visible="true">
        <AxixTitle title="">
            <Navigation></Navigation>
        </AxixTitle>
        <BarOptions barStyle="bsRectangle" barWidthPercent="70" depthPercent="100"
                    multiBar="mbNone" offsetPercent="0"/>
        <Navigation>
            VIA Object_To_Children CLASS Room
        </Navigation>
        <!--Criterion attribute="" format="">
            <Navigation></Navigation>
        </Criterion>
        <Criterion attribute="" format="">
            <Navigation></Navigation>
        </Criterion-->
        <!-- weitere Criteria -->
        <XValue attribute="Object_Display_Kontext" caption="" format=""
                operation="gopAttribute">
            <Navigation></Navigation>
        </XValue>
        <YValue attribute="bisB_NetGroundArea" caption="" format=""
               operation="gopAttribute">
            <Navigation></Navigation>
        </YValue>
    </BisSerie>

    <!-- weitere BisSerie -->

    <BisFunctionSerie bisChartType="bst_byChart" color="$000000"
                      colorAutomatic="true" drawBetweenPoints="false"
```

```
                forEachBisSerie="true" functionType="bftAverage"
                markStyle="smsValue" markVisible="false" period="0"
                periodAlign="paCenter" periodStyle="psNumPoints"
                pointerStyle="psRectangle" pointerVisible="false"
                showInLegend="false" sortXOrder="loNone" title=""
                vertAxis="aLeftAxis" visible="true">
    <BarOptions barStyle="bsRectangle" barWidthPercent="70" depthPercent="100"
                multiBar="mbNone" offsetPercent="0"/>
    <ForBisSerie name="BisSerie1"/>
    <!-- weitere ForBisSerie -->
</BisFunctionSerie>

<!-- weitere BisFunctionSerie -->

<BottomAxis automatic="true" automaticMaximum="true" automaticMinimum="true"
            axisValuesFormat="#,##0.###" dateTimeFormat="" endPosition="100"
            exactDateTime="true" gridCentered="false" horizontal="true"
            increment="0" inverted="false" labels="true"
            labelsAlign="alDefault" labelsAlternate="false" labelsAngle="0"
            labelsExponent="false" labelsMultiLine="false"
            labelsOnAxis="true" labelsSeparation="10" labelsSize="0"
            labelStyle="talAuto" logarithmic="false" logarithmicBase="10"
            maximum="0" maximumOffset="0" maximumRound="false" minimum="0"
            minimumOffset="0" minimumRound="false" minorTickCount="3"
            minorTickLength="2" otherSide="false" positionPercent="0"
            positionUnits="muPercent" roundFirstLabel="true"
            startPosition="0" tickInnerLength="0" tickLength="4"
            tickOnLabelsOnly="true" titleSize="0" visible="true"
            zPosition="0">
    <Axis color="$000000" endStyle="esRound" lineMode="lmLine" mode="pmCopy"
          smallDots="false" smallSpace="0" style="???" visible="true" width="2"/>
    <Grid color="clGray" drawEvery="1" endStyle="esRound" mode="pmCopy"
          smallDots="false" smallSpace="0" style="???" visible="true" width="2"
          zPosition="0"/>
    <LabelsFont/>
    <MinorGrid color="$000000" endStyle="esRound" mode="pmCopy"
               smallDots="false" smallSpace="0" style="???" visible="false" width="1"/>
    <MinorTicks color="clGray" endStyle="esRound" mode="pmCopy"
                smallDots="false" smallSpace="0" style="???" visible="true" width="1"/>
    <Ticks color="clGray" endStyle="esRound" mode="pmCopy" smallDots="false"
           smallSpace="0" style="???" visible="true" width="1"/>
    <TicksInner color="clGray" endStyle="esRound" mode="pmCopy"
                smallDots="false" smallSpace="0" style="???" visible="true" width="1"/>
    <Title/>
</BottomAxis>

<LeftAxis></LeftAxis>

<RightAxis></RightAxis>

<TopAxis></TopAxis>
```

```
    <Legend alignment="laRight" color="$000000" visible="false"/>
</Chart>
```

28 Report

Ermöglicht es, Crystal Reports Berichte in einem Applikationselement anzuzeigen.

28.1 Zu beachten

Wurde kein Berichtsobjekt über ein Load-Propagate oder über StartObjects in das Applikationselement reingeladen, wird auch kein Bericht ausgeführt.

28.2 Beispiel

```
<Report panel="pnRep" ID="report" autoExecute="true">
    <StartObjects>
        START INSTANCES "bisB_Report" VALUE "Name" = "Bericht: Gebäude"
    </StartObjects>
    <Contents>
        START INSTANCES "Building"
    </Contents>
</Report>
```

28.3 Attribute

Bezeichnung	Beschreibung
autoExecute	`autoExecute = <boolean>` definiert ob der Bericht automatisch gestartet werden soll, wenn das Applikationselement geladen wird. Default = "false"
executeOnSelect	`executeOnSelect = <string>` definiert ob der Bericht automatisch ausgelöst wird, sobald die Selektion ändert. Default = "true"
keepLastReport	`keepLastReport = <boolean>` definiert ob bei einer Selektion bei der kein Objekt resultiert (nach der SubContents-FilterNavigation) der angezeigte Report bestehen bleibt oder ein leeres Panel angezeigt wird. Default = "false"
onExecute	`onExecute = "showAndMaximize" | "maximize" | "doNothing"` definiert das Verhalten wenn der Bericht ausgeführt wird. `showAndMaximize:` Das Report-ApplicationElement wird eingeblendet und maximiert. `maximize:` Das Report-ApplicationElement wird maximiert (sofern minimiert), jedoch wenn ausgeblendet **NICHT** eingeblendet. `doNothing:` nichts passiert (default)

28.4 Elemente

Bezeichnung	Beschreibung

StartObjects	FilterNavigation die angibt welcher Bericht ausgeführt werden soll. Die FilterNavigation sollte höchstens **ein** (1) Objekt des Typs **bisB_Report** zurückgeben. Werden mehrere Objekte gefunden, wird das erste verwendet.
Contents	FilterNavigation die den Anfangskontext (bzw. die Anfangsselektion) des Berichts definiert. Wird verwendet wenn ein Bericht für eine gewisse Objektmenge direkt beim Laden des Applikationselementes angezeigt werden soll.

28.5 Empfangene Propagates

Bezeichnung	Beschreibung
load	Das erste Objekt der empfangenen Objektmenge wird als Report verwendet (sofern es vom Typ **bisB_Report** ist). Analog zu StartObjects.
select	Diese Objekte werden als Kontext (bzw. Selektion) für das Ausführen des Berichts verwendet. Analog zu Contents.
Execute* (custom)	Löst den Bericht aus. Dies wird normalerweise in Verbindung mit `executeOnSelect = false` verwendet um das Auslösen des Berichtes bis zu einem bestimmten Zeitpunkt zu verzögern. `Execute` wird als Prefix verstanden, so können mehrere Executes unabhängig an verschiedene Applikationselemente verschickt werden (Execute1, Execute2, ExecuteXXX).
Print* (custom)	Druckt den aktuell dargestellten Bericht aus (inkl. Drucken-Dialog).
Export* (custom)	Exportiert den aktuell dargestellten Bericht (inkl. Export-Dialog).
SetReportParam_**PARAMNAME** (custom)	Setzt den Berichtsparameter auf `Additional.Value (Formularskripting)`. Mit Hilfe dieses Propagates kann aus einem Formular z.B. ein Datums-Berichtsparameter gesetzt werden. Die Berichtsparameter werden am jeweiligen Bericht definiert. Das Präfix "SetReportParam_" wird vom Parametername gefolgt. Um also z.B. den Parameter **vonDatum** zu setzen muss das Custom-Propagate am Formular wie folgt aussehen: `<Propagate name="SetReportParam_vonDatum" operation="custom" target="reportElement"/>`
* Name wird als Prefix ausgewertet. So können z.B. mehrere Print-Propagates unabhängig voneinander verschickt werden (Print1, Print2, PrintXXX). Ein Formular kann also ein PropagateData mit Print1 und eines mit Print2 machen ohne dass beide Propagates gleichzeitig ausgelöst werden.	

29 Pages

Das Applikationselement `Pages` funktioniert ähnlich wie das Applikationselement `Tabbed`. Es ist jedoch nicht auf Basis eines Page-Controls aufgebaut, sondern kann immer nur eine Seite auf einmal anzeigen. Die aktuelle Seite wird mit Hilfe von <Condition> ausgewählt oder explizit gewählt (Operation opShowPage oder Propagates).

Ist Condition leer, dann kann die Seite immer angezeigt werden (erst ab Byron/BIS v4.11.8).

Achtung: Zeigt das ApplicationElement keine Objekte, dann ist die Condition **immer** false (mit Ausnahme obiger Zeile).

Wichtig:
Propagates müssen auf <Pages> erfolgen, welches diese dann richtig verteilt. Wenn Propagates direkt an die Elemente einer <Page> erfolgen, dann wird ein korrektes Verhalten des Applikationselements in Frage gestellt. Eine derartige Konfiguration führt auch zu einer Warnung im Log, die aber unterdrückt werden kann (vgl. Propagate).

29.1 Attribute

Attribut	Beschreibung

`default`	`default = "nameDefaultPage"` gibt den Namen der Seite an, welche angezeigt werden soll wenn durch Conditions keine Seite angezeigt werden kann.
`keepLastPage`	`keepLastPage = "true | false"` steuert das Verhalten, wenn keine oder mehrere Seiten angezeigt werden können und keine default-Seite definiert wurde. Wenn keine Seite angezeigt werden kann und kein `default` ist definiert: • `false` eine leere Seite („nichts“) wird angezeigt • `true` die zuletzt angezeigte Seite wird angezeigt Wenn mehrere Seiten angezeigt werden können (erst ab Byron/BIS v4.11.8.): • `false` die erste mögliche (`Condition` erfolgreich) Seite wird angezeigt. • `true` die zuletzt angezeigte Seite wird angezeigt, sofern sie möglich ist, sonst wie `false`. Vorgabe: `false`
`propagateAllPanels`	`propagateAllPanels = "true | false"` gibt an, ob empfangene Propagates an alle Seiten weitergeleitet werden sollen oder nur an die aktuell sichtbare. Das Propagate wird nur an das erste Applikationselement in einer Seite weitergegeben. `Default = false`

29.2 Elemente

29.2.1 Page

Definiert eine Seite und enthält folgende Unterelemente:

Element	Beschreibung
`Condition`	Bekommt als Input das über StartObjects bzw. ein Load-Propagate erhaltene Objekt. Ist die Objektmenge, welche aus dieser Filternavigation hervorgeht > 0 ist, dann wird die Seite dargestellt. Sichtbar kann jedoch nur eine Seite sein.
`SubLayout`	Siehe SubLayout.
`<Applikationselemente>`	Wenn ein `SubLayout` definiert wurde, müssen die Applikationselemente das Attribut `panel` definiert haben. Ist kein `SubLayout` definiert für die Seite, darf das Attribut weggelassen werden.

29.3 Empfangene propagates

Folgende Custom-Propagates sind definiert:

Name (`propagateName`)	Beschreibung
page	Zeigt eine benannte <Page>. Der Name der Seite muss mit dem Parameter übergeben werden. Alternativ, wenn der Parameter

	nicht gesetzt ist, kann der Index der Seite über den `code` gesetzt werden (`code` muss einen Wert zwischen 0 und der Anzahl der Seiten -1 enthalten). Beispiele:
	`<Propagate operation="custom" target="MapPages" name="map*" nameTarget="page" parameterTarget="'pgMap'"/>`
	`<Propagate operation="custom" target="MapPages" name="map*" nameTarget="page" codeTarget="1" />`
	Erst ab Byron/BIS v4.11.8.

N.B. Der Wert von *propagateName* wird case insensitive ausgewertet.
Wird `code` verwendet, dann darf der Parameter nicht gesetzt sein oder muss im Propagate zurückgesetzt werden: `parameterTarget="NULL"`.

29.4 Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
VAL_CurrentPage	Text	Name der aktuellen Seite

29.5 Beispiel

```
<Pages ID="pages" default="myDefault" panel="pnPages" propagateAllPanels="true"
       _keepLastPage="true">
    <Caption>Pages Applikationselement</Caption>
    <Page name="page1">
        <Condition>[ CLASS Location OR CLASS Room ]</Condition>
        <SubLayout orientation="horizontal">
            <Panel ID="pnGrid2"/>
            <Panel ID="pnGrid3"/>
        </SubLayout>
        <Grid ID="grid2" panel="pnGrid2">
            <Propagate operation="load" target="grid3">
                <Transform>START INSTANCES Space</Transform>
            </Propagate>
        </Grid>
        <Grid ID="grid3" panel="pnGrid3">
            <Propagate operation="select" target="pages"/>
        </Grid>
    </Page>
    <Page name="myDefault">
        <Condition>BLOCK</Condition>
        <Grid>
            <Caption>DEFAULT</Caption>
        </Grid>
    </Page>
    <Page>
        <Condition>[ CLASS Building OR CLASS Person ]</Condition>
        <Tree ID="tree2"/>
    </Page>
</Pages>
```

30 PickablePicture (erst ab Byron/BIS v4.9.3)

Das Applikationselement PickablePicture ermöglicht es, ein Bild, mit Hilfe eines Pfades oder einer URI (base64), darzustellen. Einzelne Bereiche können "pickbar", also anwählbar sein um die Selektion zu propagieren.

30.1 Elemente

30.1.1 Picture

Definiert, welches Bild im Applikationselement angezeigt werden soll.

Attribut / Textinhalt	Beschreibung
`bisAttribute`	Enthält ein `Byron/BIS-Attribut`, welches entweder: • Den Pfad des Bildes für das geladene Objekt enthält **oder** • eine URI welche das Bild base64-Kodiert enthält
`<Textinhalt>`	Enthält den Pfad des darzustellenden Bildes. Dieser Wert wird auch als "Fallback" verwendet, wenn das in `bisAttribute` angegebene Attribut keinen Wert enthält.

30.1.2 PickableObjects

Definiert, welche Bereiche im Bild anwählbar sind (und auch ein Propagate auslösen).

Attribut / Textinhalt	Beschreibung
`mapAreaAsTextAttribute`	Enthält ein `Byron/BIS-Attribut`, welches 2 Koordinaten (2 Ecken eines Rechtecks) in folgender Form enthalten: `X1, Y1, X2, Y2` - Bsp.: `135,141,236,234`
`<Textinhalt>`	Enthält die FilterNavigation zu den anwählbaren Objekten. Diese Objekte müssen das `Byron/BIS-Attribut` welches in `mapAreaAsTextAttribute` verwendet wird erlauben.

30.2 Empfangene Propagates

Bezeichnung	Beschreibung
load	Das erste Objekt der empfangenen Objektmenge wird für das `Picture`-Element verwendet.
select	Diese Objekte werden im Bild "selektiert" (optisches Feedback auf dem Bild).
ZoomPercentage* (custom)	Die in Data übergebene Zahl wird als Zoom-Prozentzahl verwendet.
* Name wird als Prefix ausgewertet. So können z.B. mehrere Print-Propagates unabhängig voneinander verschickt werden (Print1, Print2, PrintXXX). Ein Formular kann also ein PropagateData mit Print1 und eines mit Print2 machen ohne dass beide Propagates gleichzeitig ausgelöst werden.	

30.3 Ausgelöste Propagates

Das Selektieren von anwählbaren Bereichen (definiert durch das `PickableObjects`-Element) löst ein Propagate aus.

30.4 Beispiel

```
<PickablePicture ID="pickPict" panel="pniBrowser">
    <Contents>CLASS "bisB_AnImport"</Contents>
    <Picture bisAttribute="bisB_PictureSource"/>
    <PickableObjects mapAreaAsTextAttribute="bisB_HTMLAreaCoords">
        VIA "Object_To_Children"
    </PickableObjects>
    <Propagate operation="load" target="myForm"/>
    <Propagate operation="select" target="gridDevices"/>
</PickablePicture>
```

31 Toolbar (erst ab Byron/BIS v4.8.9)

Das Application Element Toolbar (ToolbarElem) ermöglicht die Platzierung von Toolbars an beleibigen Orten im Layout. Ein ToolbarElem enthält beliebig viele Toolbars. Die Toolbars können jedoch nicht verschoben und ausgeblendet werden.

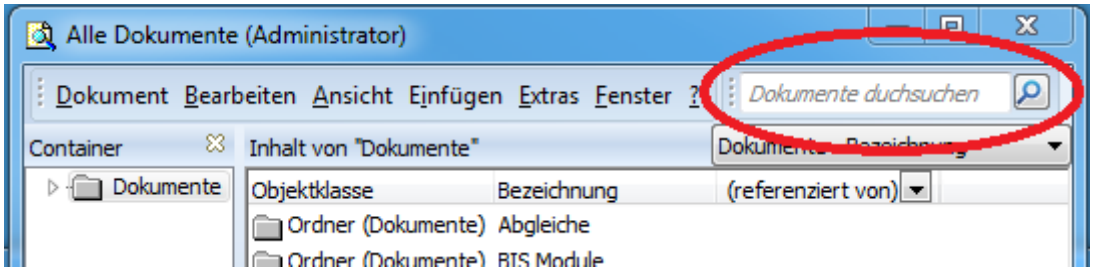
Definition der einzelnen Toolbars analog Toolbar

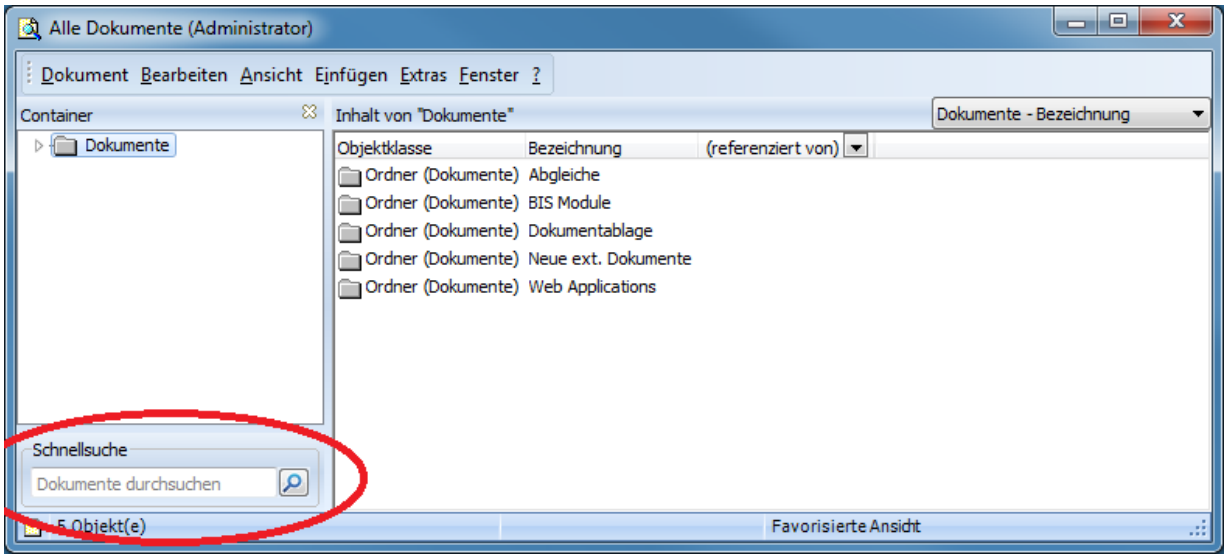
31.1 Beispiel

```
<ToolbarElem ID="toolbar" panel="pnToolbar">
    <Toolbar>
        <ToolbarButton><Execute name="Undo"/></ToolbarButton>
        <ToolbarSeparator/>
        <ToolbarButton><Execute name="CutClipboard"/></ToolbarButton>
        <ToolbarButton><Execute name="CopyClipboard"/></ToolbarButton>
        <ToolbarButton><Execute name="Paste"/></ToolbarButton>
        <ToolbarButton><Execute name="Delete"/></ToolbarButton>
        <ToolbarSeparator/>
        <ToolbarButton><Execute name="TreeSearch"/></ToolbarButton>
    </Toolbar>
</ToolbarElem>
```

32 QuickSearch (erst ab Byron/BIS v4.10.2)

Das (unsichtbare) Application Element QuickSearch steuert die QuickSearch-Toolbar, bzw. das QuickSearch-ApplicationElement.





Nach dem Klicken der Lupe oder Eingabe von <Enter> wird die Navigation des Elements `Search` ausgeführt und die Ergebnisse werden propagiert.

32.1 Beispiel

Im folgenden Beispiel werden über die Suchnavigation alle Dokumente gesucht, deren Bezeichnung den Suchtext enthält. Damit die Caption am Grid „Suchergebnisse" anzeigt, wird ein entsprechendes Custom-Propagate nachgeschickt.

```
<QuickSearch ID="quickSearch" sendSelectNotification="true" emptyText="Dokumente suchen">
    <Search>
        -- Alle Dokumente deren Namen den Suchtext enthält
        START INSTANCES Doc_Document
        VALUE Name ~= '="*" + VAL_SearchText + "*"'
    </Search>
    <Propagate operation="load" target="grid"/>
      <Propagate operation="custom" target="grid" nameTarget="caption"
                 parameterTarget="'@Frmwrk_CaptionSearchResults@'" />
</QuickSearch>
```

Und die dazugehörige Konfiguration des Grids:

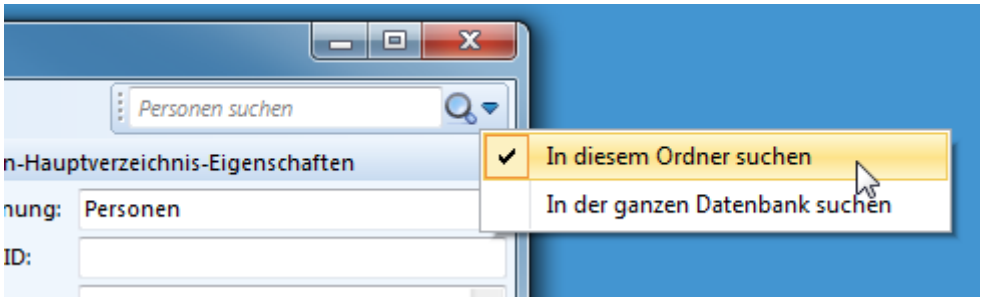
```
<Grid ID="grid" panel="pnGrid" onDoubleClick="Open" containerSupport="true">
    <Contents>
        IF "VAR('quickSearch.VAL_IsPropagating')" (
          PASS -- Ergebnis von QuickSearch
        ELSE
          RECALL inputObjects VIA Object_To_Children
        )
        CLASS Doc_Classes
    </Contents>
    <ContentsInverse>VIA "Object_To_Parent"</ContentsInverse>
    <GridView alwaysPrefer="true"/>
    <StatusMapping>
        <StatusMapEntry ID="GridContents" panel="1"/>
        <StatusMapEntry ID="GridAutoSum" panel="2"/>
        <StatusMapEntry ID="GridPreferredView" panel="3"/>
    </StatusMapping>
</Grid>
```

Damit die Caption des Grids nach einem Laden durch Selektieren im Baum (wieder) korrekt dargestellt wird, sendet der Baum **vor** dem Laden des Grids ein entsprechendes Custom-Propagate an das Grid:

```
<Tree ID="tree" …>
    …
    <Propagate operation="custom" target="grid" nameTarget="caption"
               parameterTarget="'@Frmwrk_GridCaption@'" />
    <Propagate operation="load" target="grid"/>
    …
</Tree>
```

32.2 Mehrere Suchen (ab v5.0.0)

Durch Angabe von mehreren Elementen `<Search>` unterstützt QuickSearch gleichzeitig mehrere Suchen. Damit können z. Bsp. eine lokale und eine globale Suche mit einem einzigen QuickSearch realisiert werden. Die Auswahl erfolgt durch das Dropdown neben dem Lupensymbol:



Die Einstellungen werden von <QuickSearch> übernommen, können aber in vielen Fällen für jede Suche überschrieben werden.

```
   <QuickSearch ID="quickSearch" emptyText="@stamm_PersonSearch@" >
      <Search caption="In diesem Ordner suchen" >
          RECALL "tree.OBJ_Selection"
          CLASS bisB_PersonContainer
          [
```

```
        OR
          LOOP (
            VIA Object_To_Children
          )
        ]
        VALUE Object_Display_Kontext ~= "='*' + VAL_SearchText + '*'"
    </Search>
    <Search caption="In der ganzen Datenbank suchen"
            emptyText="@stamm_PersonSearch@ (global)">
        START INSTANCES Person
        VALUE Object_Display_Kontext ~= "='*' + VAL_SearchText + '*'"
    </Search>
    …
</QuickSearch>
```

Hinweis: Mehrere Suchen werden nicht unterstützt, wenn QuickSearch auf einem Panel platziert wird!

32.3 Attribute

Bezeichnung	Beschreibung
autoSearch	`autoSearch = <boolean>` Optional. Definiert ob die Suche automatisch ausgelöst wird, dh. die Suche wird 0.5s nach dem Drücken des letzten Zeichens ausgelöst. Default = "false" Ab Byron/BIS v4.10.4 Auch für die Search-Elemente erlaubt (ab Byron/BIS v5.1.2).
caption	Bestimmt den Eintrag im Auswahlmenü der Suchen. Nur für die Search-Elemente erlaubt.
emptyText	`emptyText = <string>` Optional. Definiert den Text, welcher im Hintergrund angezeigt wird ("Wasserzeichen"). Default = "Objekte suchen" Auch für die Search-Elemente erlaubt (ab Byron/BIS v5.0.0).
formula	`formula = <string>` Optional. Definiert die Darstellung der Objekte in $-Notation in der Popup-Liste. Nur in Verbindung mit `singleResult="true".` Default = „$Object_Display_Kontext$ Ab Byron/BIS v5.0.0
ID	Identifikation des QuickSearch-Elements bzw. der Suchen (Search-Elemente). Die Identifikationen der Suchen müssen nicht eindeutig sein. Auch für die Search-Elemente erlaubt (ab Byron/BIS v5.1.2). Vgl. Parameter für die FilterNavigation

maxItemsDisplayed	`maxItemsDisplayed = <integer>` Optional. Definiert wieviel Objekte maximal in der Popup-Liste angezeigt werden. Nur in Verbindung mit `singleResult="true"`. Default = 20 Ab Byron/BIS v5.0.0
minChars	`minChars = <integer>` Optional. Definiert wie viele Zeichen eingegeben werden müssen, damit die Suche ausgeführt werden kann. Default = "0" (bis Byron/BIS v4.11.8) Default = "1" (bis Byron/BIS v5.0.0) Ab Byron/BIS v4.10.4 Auch für die Search-Elemente erlaubt (ab Byron/BIS v5.0.0).
panel	`panel = <string>` Optional. Name des Panels, auf welchem das Application Element erscheinen soll. Ohne Angabe des Panels wird QuickSearch als unsichtbares ApplicationElement verwendet (QuickSearch-Toolbar oben rechts im Fenster). Hinweis: Am zugehörigen Layout, bzw. Panel, dürfen minHeight und MaxHeight nicht gesetzt sein! Ab Byron/BIS v4.10.4
resetVisible	`resetVisible = <boolean>` Optional. Definiert die Sichtbarkeit des Reset-Buttons. Wird der Reset-Button gedrückt, wird ein Propagate ausgelöst und der FilterNav-Parameter "`VAL_SearchResetting`" ist `true`, bis eine erneute Eingabe getätigt wird. Ab Byron/BIS v5.4.9
singleResult	`singleResult = <boolean>` Optional. Definiert ob QuickSearch höchstens ein Objekt propagieren soll. Bei einem Suchergebnis von mehr als einem Objekt wird eine Auswahl angezeigt. autoSearch wird bei `singleResult="true"` deaktiviert. Default = "false" Ab Byron/BIS v5.0.0 Auch für die Search-Elemente erlaubt (ab Byron/BIS v5.1.2).
width	`width = <integer>` Optional. Definiert die Breite des QuickSearcheingabefeldes. Default = 157, Minimum = 100, Maximum = 300 Ab Byron/BIS v5.0.0

32.4 Empfangene Propagates

QuickSearch reagiert auf folgende Propagates:

- `<Propagate operation="load" …"/>`, bzw.
  `<Propagate operation="custom" nameTarget="Reset" …/>`
  Der Suchtext (Editor in der Toolbar) wird zurückgesetzt (ab Byron/BIS v4.10.4).
- `<Propagate operation="custom" nameTarget="Execute" …/>`
  Die Suche wird ausgeführt (ab Byron/BIS v4.10.4).
  Mit dem Attribut `codeTarget` wird gesteuert, ob die Suche auch ausgeführt werden soll, wenn kein Suchtext vorhanden ist. `codeTarget="0"`: Suche immer ausführen, `codeTarget="1"`: Suche nur ausführen, wenn Suchtext vorhanden.
  Hinweis: Bei mehreren Suchen wird die zuletzt verwendete Suche ausgeführt.

32.5 Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
`VAL_CurrentQuickSearch`	Text	ID der aktuellen Suche Ab Byron/BIS v5.1.2
`VAL_SearchResetting`	Boolean	True, wenn die Suche über den Reset-Button zurpckgesetzt wordem ist. Ab Byron/BIS v5.4.9

32.6 Tipps:

- Für die Verwendung einer bestimmten Ansicht für die Darstellung der Schnellsuche diese Ansicht auf das Aktions-Objekt favorisieren und `<GridView alwaysPrefer="true"/>` setzen.
- Damit erneutes Klicken auf Baum den selektierten Knoten wieder propagiert:
  `<Tree … propagateOnFocusChanged="true">`
- Damit "Zeigen" mittels Drag-Drop korrekt funktioniert, dh. der "richtige Ordner" im Baum geöffnet wird, muss am entsprechenden Grid ein `<ContentsInverse>` konfiguriert werden (z. Bsp. `<ContentsInverse>VIA "Object_To_Parent"</ContentsInverse>`) und das Attribut `dropShowTryOthersFirst="true"` gesetzt sein.
- Wird die Suche zurückgesetzt (`resetVisible = "true"`), muss je ein Custom-Propagate und ein Load-Propagate ans Grid erfolgen, damit die Grid-Beschriftung und die Grid-Objekte wieder geladen werden.
  Weiter ist unter `<Grid><Contents>` die Filternavigation zu erweitern für die Ergebnisse aus QuickSearch (z.B `«IF "VAR('quickSearch.VAL_IsPropagating') AND (NOT VAR('quickSearch.VAL_SearchResetting'))" (… »`.
  Im ModulMaster in den Stammdaten ist das Ganze für Personen und Firmen umgesetzt worden.

33 Lookup (erst ab Byron/BIS v4.10.4)

Das Application Element Lookup ermöglicht die Verwendung einer einzelnen BisLookupCombobox in einem Werkzeug.

33.1 Beispiel

```
<Lookup ID="lookup" panel="myPanel" emptyText="Meine Objekte suchen"
        showImage="true" allowDrag="true">
    <Search>
        START VALUE bisP_MessageNo ~= "='*' + VAL_SearchText + '*'"
    </Search>
    <Propagate operation="load" target="grid"/>
</Lookup>
```

34 Map (ab Byron/BIS v4.11.1)

Das Applikationselement Map ermöglicht es, Objekte auf einer Karte oder auf einem Satellitenbild anzuzeigen. Mit der linken Maustaste auf der Karte kann diese verschoben werden. Durch drücken der Ctrl- oder Shift Taste wird ein ‚Box-Pick' initiiert. Durch Drücken auf einem selektierten Objekt wird ein Drag initiiert.

34.1 Beispiel

```
<Map ID="MapID" panel="MapPanel" center="7.61 47.5" zoom="12">
    <MapService style="OSM" url="http://a.tile.openstreetmap.org/" clear="false"/>
    <Color default="#FFFF00" alpha="100">
        START INSTANCES BisG_FAinstance VALUE Name = "Strassenstücke"
    </Color>
    <Selection mode="outline" color="#FF0000" width="2.5"/>
    <!-- Propagates -->
    <Propagate operation="load" target="grid" name="NewNotification"/>
    <Propagate operation="load" target="grid">
        <Transform>
            VIA "demo_spaceToOrders"
        </Transform>
    </Propagate>
    <StatusMapping>
        <StatusMapEntry ID="MapObject" panel="3"/>
    </StatusMapping>
</Map>
```

34.2 Attribute

Bezeichnung	Beschreibung
switchWidth (ehem. backgroundWidth) ab V5.1.0	Breite des Ansichtsknopfes. Dieser wird eingeblendet, wenn mehrere MapService und oder Färbungen definiert werden. Vorgabewert: 180
center	Voreingestelltes Kartenzentrum in WGS84 (Dezimal, X und Y durch Leerschlag getrennt). Das Applikationselement merkt sich das aktuelle Kartenzentrum in der Registry und verwendet dieses beim wiederholten Aufstarten. Vorgabewert: "8.3 47.05"
gotoZoom (ab v4.11.8 / v4.11.7.2208)	Grösster Zoomfaktor, der beim Zeigen eingestellt wird, wenn die Objekte darin Platz haben. Vorgabewert: maximum - 2
legendMargin (ab v4.11.8 / v4.11.7.2208)	Abstand der Legende vom linken Rand. Vorgabewert: backgoundWidth
legendWidth (ab v4.11.8 / v4.11.7.2208)	Breite der Legende. Vorgabewert: 180

maximum	Maximaler Zoomfaktor. Vorgabewert: 20
minimum	Minimaler Zoomfaktor. Vorgabewert: 4
mode	"solid", "hatch" oder "outline" default solid. Definiert die Darstellungsart der Flächen (contentsShape)
saveZoomCenter	"true": Das Applikationselement merkt sich den aktuellen Zoomfaktor und Bildzentrum in der Registry und verwendet diese Werte beim wiederholten Aufstarten. "false": Es gelten die Werte von "zoom" und "center" Vorgabewert: true
zoom	Voreingestellter Zoomfaktor zwischen 2 und 18. Bei saveZoomCenter="true" wird dieser Wert ev. überschrieben. Vorgabewert: 10

34.3 Element ContentsShape, ContentsIcon, ContentsLine

Diese Elemente (beide dürfen mehrfach vorkommen) enthalten Filternavigationen, welche die darzustellenden Objekte definieren. (ContentsIcon kann auch Elemente enthalten siehe unten)

Die Eingangsobjekte sind die Resultatobjekte der Contents-Navigation, falls eine solche definiert ist.

`<ContentsShape>` definiert die Flächenobjekte, d.h. es wird das Attribut `bisB_AreaGeometrie` ausgewertet. `<ContentsIcon>` zeichnet ein 16x16 oder 32x32 Icon and die Position des Attributes `Position`.

Attribute von ContentsShape / ContentsIcon

Bezeichnung	Beschreibung
maximum	Maximaler Zoomfaktor, damit die Objekte gezeigt werden
minimum	Minimaler Zoomfaktor, damit die Objekte gezeigt werden
name (nur ContentsIcon)	Name des Icons das verwendet wird. Wenn der Name nicht gesetzt wird, dann wird das Icon des Objektes genommen. $-Ausdrücke werden ausgewertet.
size (nur ContentsIcon)	Definiert ob die grossen oder kleinen Icons verwendet werden (Mögliche Werte: 16 oder 32) Vorgabewert: 32.

Elemente in ContentsIcon ab Byron/BIS V4.11.8 / 4.11.7.2208

<Navigation> enthält Filternavigationen, welche die darzustellenden Objekte definiert.Eingangsobjekte sind die Resultatobjekte der Contents-Navigation, falls eine solche definiert ist.

<PositioningCondition> Bedingung, die geprüft wird, wenn ein Objekt positioniert wird. Eingangsobjekt ist jeweils das zu platzierende Objekt.

<AfterPositioning> Diese Filternavigation wird nach dem platzieren (setzen des Attribut 'Position') noch in derselben Schreibtransaktion ausgeführt.

Zusätzliche Attribute von ContentsLine ab Byron/BIS V5.3.2

Für diese Objekte wird das Attribut `bisB_Jointer` ausgewertet.

www.byron.ch/downloads/dokumente/ByronBIS_Jointer.pdf

Die Bedeutung der Breitenvorzeichen und die Attributvorgaben entsprechen dieser Dokumentation

Bezeichnung	Beschreibung
alpha	Transparenz als Zahl im Bereich 0…255 0 durchsichtig … 255 undurchsichtig (opak) Wird ignoriert bei colorValue="byElement"
colorValue	Farbe die verwendet wird, zB. "#77E6E5" Somit dürfen die Attribute colorDefault und colorAttribute nicht definiert sein. Der Wert kann "byElement" sein, damit wird die Färbung aus dem Element <Color> bestimmt und es erfolgt ein Legendeneintrag.
colorDefault	Farbe die verwendet wird, wenn das Attribut undefiniert ist. zB. "#77E6E5"
colorAttribute	Attribut welches die Farbe definiert. Vorgabewert: `bis_Color`
style	Linienart. "Solid", "Dash", "Dot", "DashDot" Vorgabewert. "Solid"
widthValue	Breite (Dicke) die verwendet wird. zB. "-3" entspricht 3 Pixel.
widthDefault	Breite die verwendet wird, wenn das Attribut undefiniert ist. zB. "0.2" (20 cm)
widthAttribute	Attribut welches die Breite definiert. Vorgabewert: `bisG_Width`

Beispiel:

```
<Contents>
    [START INSTANCES "Building" OR START INSTANCES "bisAR_OpenArea" ]
</Contents>

<ContentsShape minimum="12">
    PASS
</ContentsShape>

<ContentsIcon minimum="14" maximum="18" size="32">
    VIA Room_To_Component
</ContentsIcon>

<ContentsLine alpha="80" colorDefault="#776655" widthDefault="-3" >
   START INSTANCES bisB_Cable
</ContentsLine>
```

```
<ContentsIcon size="16" minimum="16">
  <Navigation>START '{89F54518-5CE9-4A6C-9649-78D1DCE86D17}' VIA Object_To_Children
  </Navigation>
  <PositioningCondition>CLASS bisP_Signal
  </PositioningCondition>
  <AfterPositioning>
    COUNT (VIA Room_of_Component) = 0
    BIND Room_of_Component (START '{48BEB6D8-86A2-434A-8A01-0E80726AD097}' ) REBIND
  </AfterPositioning>
</ContentsIcon>
```

34.4 Ansicht

Es kann zwischen verschiedenen Ansichten umgeschaltet werden, wenn es mehr als eine Färbung (Element Color) oder mehr als einen Hintergrund (Element MapService) gibt (ab V5.1.0). Die Breite des Knopfes kann mit switchWidth definiert werden. Die Titel sind durch @Frmwrk_View@, @Frmwrk_Background@ und @Frmwrk_Coloring@ definiert.

34.5 Element Color

Definiert die Farben der Flächen mit Hilfe einer Färbung oder "fix" bzw. "default". Die gewünschte Färbung muss mit einer Filternavigation identifiziert werden.

Verwendung einer Färbung:

```
<Color default="#FFFF00" alpha="100" caption="@street@" >
    START INSTANCES BisG_FAinstance VALUE Name = "Strassenstücke"
</Color>
```

Es könnnen mehrere <Color> Elemente definiert werden (ab V5.1.0) Die Filternavigation darf auch mehrere Färbungen ergeben. In diesem Fall kann in der Ansicht zwischen den verschiedenen Färbungen umgeschaltet werden. Mit dem Attribut caption kann der Name der Färbung überschrieben werden. Im mode "solid" gewählt, wird die Transparenz durch den Wert von *alpha* festgelegt. Ein Alphawert von 255 bedeutet undurchsichtig / opaque.

34.6 Element MapService

Definiert welcher Provider für den Kartenhintergrund verwendet werden soll. Wird kein MapService definiert, dann wird der *style* **OSM** mit den Vorgabeeinstellungen verwendet.

Beispiel:

```
<MapService caption="Standard" style="OSM" url="http://a.tile.openstreetmap.org/"
clear="true"/>
```

Im Folgenden werden die möglichen Werte für *style* sowie die zugehörige Vorgabe für *url* angegeben**. N.B**. der Wert von style ist case insensitive.

style	url Vorgabe	Beschreibung
BingSatellite	http://ecn.t%d.tiles.virtualearth.net/tiles/	Satellitenbilder von bing.
BingMap	http://ecn.t%d.tiles.virtualearth.net/tiles/	Karten von bing.
BingHybrid	http://ecn.t%d.tiles.virtualearth.net/tiles/	Hybridbilder (Satellit/Karte) von bing.
WMS	-	WebMapService Dienst.

WMT	-	WebMapTileService der google Kachelnummern Schema unterstützt.
OSM	http://a.tile.openstreetmap.org/	OpenStreetMap – Karten. Siehe https://wiki.openstreetmap.org/wiki/Tile_servers für mögliche URL Verwendungen. **Wichtig:** "/" am Ende nicht vergessen. Beispiel: http://a.tile.openstreetmap.org/
OSMStatic	http://staticmap.openstreetmap.de/	OpenStreetMap – Karten.

Mit dem Attribut *url* kann die Vorgabe überschrieben werden. Im Text kann auch ein globaler Parameter oder eine Umgebungsvariable verwendet werden (ab v5.1.0).

Beispiel:

```
<MapService caption="Standard" style="WMS" url="http://%OurWMSServer%" clear="true"/>
```

Das Attribut *clear* gibt an, ob beim Laden der Konfiguraion der Tile-Cache im %TEMP% Verzeichnis des Benutzers gelöscht werden soll. Ab Version 5.3.0 kann auch das maximale Alter einer Cache-Datei spezifiziert werden (In Tagen)

Beispiele:

`<MapService clear="true">`	Cache wird immer gelöscht.
`<MapService clear="false">`	Cache wird nie gelöscht.
`<MapService clear="5">`	Cachedateien älter als 5 Tage werden gelöscht.
`<MapService clear="0.125">`	Cachedateien älter als 3 Stunden werden gelöscht.

WMT-Kartenprovider ab Byron/BIS V5.3.3

Für WTS-Kartenprovider müssen *url* und ein *name* angegeben werden. Der url enthält Platzhalter den ZoomLevel (%0:d), x- (%1:d) und y Kachelnummer (%2:d). Diese Werte enttsprechen der Google Kachelnummerierung. Der name ergibt den Namen des cache Ordners.

```
<MapService caption="Open Street" style="WMT" name="WMTOST"
   url="http://a.tile.openstreetmap.org/[zoom]/[x]/[y].png" />
```

Ergibt z.B eine Kachel 2142_1439.png in %tmp%\kacheln\OST\level_12

Dies entspricht http://a.tile.openstreetmap.org/12/2142/1439.png

N.B. weder .png am Schluss noch die Reihenfolge von x und y sind Pflicht, am Schluss muss der URL stimmen.

WMS-Kartenprovider

Für WMS-Kartenprovider müssen *url* und die gewünschten Layer angegeben werden. Die Layer werden als Text des Elements MapService konfiguriert. Die einzelnen Layer sind durch Kommas getrennt. Für WMS kann auch das Koordinatensystem (srs) und reaspect definiert werden.
Mögliche Werte von srs: http://spatialreference.org/ref/epsg/21781/

"EPSG:4326" WGS84 (default)
"EPSG:21781" CH1903
"EPSG:3857" auch bekannt als EPSG:900913 ab Version 4.11.7
"EPSG:2056" CH1903+ / LV95 ab Version 5.3.2

reaspect ist "TRUE", "FALSE" oder undefiniert zur Bedeutung:
http://webhelp.esri.com/arcims/9.3/general/mergedprojects/wms_connect/wms_connector/get_map.htm

Der MapService kann Layers und Styles Elemente beinhalten. ab Byron/BIS V4.11.8 / 4.11.7.2208

```
<MapService >Luftbild,EKS</MapService>
```

Entspricht

```
<MapService >
  <Layers>Luftbild,EKS</Layers>
  <Styles>default</Styles>
</MapService>
```

Dies führt in ArcGIS zum Fehler:

Parameters 'styles' and 'layers' should have the same number of values.

Also gleichviele Felder in Styles angeben oder gar kein Styles, das akzeptiert auch ArcGis. Da ein Weglassen zu obigem default führt, muss also ein leeres Styles explizit angeben werden:

```
<MapService >
  <Layers>Luftbild,EKS</Layers>
  <Styles/>
</MapService>
```

Wenn mehrere MapServices definiert werden, dann wird ein Schalter generiert, welcher das Umschalten zwischen den verschiedenen Karten ermöglicht. ab Byron/BIS V4.11.8 / 4.11.7.2208

Wenn Layernamen Leerzeichen enthalten, dann müssen diese durch %20 ersetzt werden. ab Byron/BIS V5.1.2

Beispiele:

```
<MapService caption="Swisstopo" style="WMS" url="http://wms.geo.admin.ch/">
    ch.swisstopo.images-landsat25,
    ch.swisstopo.swissboundaries3d-gemeinde-flaeche.fill,
    ch.swisstopo.swissboundaries3d-bezirk-flaeche.fill,
    ch.swisstopo.swissboundaries3d-land-flaeche.fill,
    ch.swisstopo.swissboundaries3d-kanton-flaeche.fill
</MapService>
```

```
<MapService caption="Mapwerks"  style="WMS"
            url="http://sidp.mapwerks.com/cubeserv/cubeserv.cgi">
    Foundation.depthl_1m:CubeWerx, Foundation.polbndl_1m:CubeWerx,
    Foundation.inwatera_1m:CubeWerx, Foundation.watrcrsl_1m:CubeWerx,
    Foundation.builtupa_1m:CubeWerx, Foundation.railrdl_1m:CubeWerx,
    Foundation.roadl_1m:CubeWerx, Foundation.treesa_1m:CubeWerx
</MapService>
```

```
<MapService caption="Arc-GIS"   style="WMS" url="http://my/arcgis/WMSServer"
    srs="EPSG:21781" >
  <Layers>Parzelle,Eigentuemer<Layers>
  <Styles/>
</MapService>
```

```
<MapService caption="Arc-GIS"   style="WMS" url="http://my/arcgis/WMSServer"
    srs="EPSG:21781" >
  <Layers>Luftbild,EKS<Layers>
  <Styles>default,default<Styles>
</MapService>
```

Aus diesen Parametern wird der URL aufgebaut:

<ServerURL>?version=1.1.1
&service=WMS&REQUEST=GetMap
&STYLES=<StyleList>
&SRS=EPSG:21781 bzw. Wert des srs Attributes
&Layers=<Layerlist>
&REASPECT=TRUE gem. Attribut reaspect= "true" / "false"
&BBox=minX,minY,maxX,maxY
&Width=256
&Height=256
&FORMAT=image/png

Die URL des ersten Beispiels hat damit beispielhaft diese Form:

http://wms.geo.admin.ch/?version=1.1.1&service=WMS&REQUEST=GetMap&STYLES=default
&SRS=EPSG:4326&Layers=ch.swisstopo.images-landsat25,ch.swisstopo.swissboundaries3d-gemeinde-flaeche.fill,ch.swisstopo.swissboundaries3d-bezirklaeche.fill, ch.swisstopo .swissboundaries3d-land-flaeche.fill, ch.swisstopo.swissboundaries3d-kanton-flaeche.fill
&BBox=6.32812499999999,47.9899216674142,7.03124999999999,48.4583518828087
&WIDTH=256&HEIGHT=256&FORMAT=image/png

Im Debug-Level wird die URL in die Logdatei ausgegeben, falls das Bild nicht im cache gefunden wird.

34.7 Element Selection

Definiert wie die Selektion dargestellt werden soll. Dazu werden die XML-Attribute *mode*, *color*, *alpha*, *style* und *width* verwendet. Die Attribute *color* und *alpha* definieren in allen Fällen die Farbe der Selektion Die Vorgabe ist rot (#FF0000) und opaque (255). Die Verwendung der anderen Attribute hängt von dem gewählten Wert von *mode* ab. Die folgende Tabelle beschreibt die möglichen Werte für *mode*.

Bezeichnung	Beschreibung
solid	Die selektierte Fläche wird einfarbig mit der angegebenen Farbe gezeichnet.
hatch	Die selektierte Fläche wird mit einem Muster gezeichnet. Das Muster wird über das Attribute *style* – einer Zahl zwischen 0 und 53 gewählt. Die Vorgabe für *style* ist 52.

	HatchStyleHorizontal = 0, HatchStyleVertical = 1, HatchStyleForwardDiagonal = 2, HatchStyleBackwardDiagonal = 3, HatchStyleCross = 4, HatchStyleDiagonalCross = 5, HatchStyle05Percent = 6, HatchStyle10Percent = 7, HatchStyle20Percent = 8, HatchStyle25Percent = 9, HatchStyle30Percent = 10, HatchStyle40Percent = 11, HatchStyle50Percent = 12, HatchStyle60Percent = 13, HatchStyle70Percent = 14, HatchStyle75Percent = 15, HatchStyle80Percent = 16, HatchStyle90Percent = 17, HatchStyleLightDownwardDiagonal = 18, HatchStyleLightUpwardDiagonal = 19, HatchStyleDarkDownwardDiagonal = 20, HatchStyleDarkUpwardDiagonal = 21, HatchStyleWideDownwardDiagonal = 22, HatchStyleWideUpwardDiagonal = 23, HatchStyleLightVertical = 24, HatchStyleLightHorizontal = 25, HatchStyleNarrowVertical = 26, HatchStyleNarrowHorizontal = 27, HatchStyleDarkVertical = 28, HatchStyleDarkHorizontal = 29, HatchStyleDashedDownwardDiagonal = 30, HatchStyleDashedUpwardDiagonal = 31, HatchStyleDashedHorizontal = 32, HatchStyleDashedVertical = 33, HatchStyleSmallConfetti = 34, HatchStyleLargeConfetti = 35, HatchStyleZigZag = 36, HatchStyleWave = 37, HatchStyleDiagonalBrick = 38, HatchStyleHorizontalBrick = 39, HatchStyleWeave = 40, HatchStylePlaid = 41, HatchStyleDivot = 42, HatchStyleDottedGrid = 43, HatchStyleDottedDiamond = 44, HatchStyleShingle = 45, HatchStyleTrellis = 46, HatchStyleSphere = 47, HatchStyleSmallGrid = 48, HatchStyleSmallCheckerBoard = 49, HatchStyleLargeCheckerBoard = 50, HatchStyleOutlinedDiamond = 51, HatchStyleSolidDiamond = 52, HatchStyleTotal = 53
outline	Die selektierte Fläche wird zusätzlich in der gewählten Farbe umrandet. Die Dicke des Randes wird mit dem Attribut *width* festgelegt. Die Vorgabe für *width* ist 3.0.

Beispiele:

```
<Selection mode="solid" color="#FF0000" alpha="200"/>
<Selection mode="hatch" color="#FF0000" style="27"/>
<Selection mode="outline" color="#FF0000" width="2.5"/>
```

34.8 Propagates

34.8.1 Custom Propagates (Input)

Folgende Custom-Propagates (in) sind definiert.

Name (`propagateName`)	**Beschreibung**
center	Verschiebt das Kartenzentrum in WGS84-Koordinaten auf den im Parameter angegeben Wert. Beispiel vgl. unten.
zoom	Ändert den Zoomfaktor auf den im Parameter angegebenen Wert. Zulässige Werte liegen zwischen 2 und 18. Beispiel vgl. unten.

zoomCenter	Verschiebt das Kartenzentrum und ändert den Zoomfaktor (eine Kombination der obigen Operationen). Der Parameterwert enthält zuerst den Zoomfaktor und danach die Koordinate. Beispiel: `'13 8.2357 47.4436'`
clear	Löscht den Tile-Cache im %TEMP% Verzeichnis des Benutzers. Der Parameter *fromParameter* wird hier nicht verwendet.

N.B. Der Wert von *propagateName* wird case insensitive ausgewertet.

Beispiele mit opPropagate:

```
<Operation ID="BirrfeldCenter" caption="Birrfeld" opCode="opPropagate" toElement="Map"
           propagateType="custom" propagateName="center" fromParameter="'8.2357 47.4436'"/>
    <!-- Wert in einfachen '' -->
<Operation ID="BirrfeldZoom" caption="Birrfeld" opCode="opPropagate" toElement="Map"
           propagateType="custom" propagateName="zoom" fromParameter="'13'"/>
<!-- Wert in einfachen '' -->
```

```
<Operation ID="MapClear"
      caption="Cache löschen"
       opCode="opPropagate"
toElement="Map"
propagateType="custom"
propagateName="clear"
fromParameter="-"/> <!-- nicht verwendet -->
```

```
<Operation ID="BirrfeldZC"
      caption="Birrfeld"
       opCode="opPropagate"
    toElement="Map"
propagateType="custom"
propagateName="zoomCenter"
fromParameter="'13 8.2357 47.4436'" /> <!-- Wert in einfachen '' -->
…
```

```
<Menu>
…
<Item><Execute name="BirrfeldZoom"/><Execute name="BirrfeldCenter"/></Item>
<!-- Zoom und Verschieben des Kartenzentrums wird hier kombiniert -->
<Item><Execute name="MapClear"/></Item>
<Item><Execute name="BirrfeldZC"/></Item>
…
</Menu>
```

34.8.2 Custom Propagates (Output)

Folgende Custom Propagates werden vom Applikationselement ausgelöst.

Name (`propagateName`)	Beschreibung
center	Propagiert das Kartenzentrum in WGS84-Koordinaten. Beispiel: "7.5 47.85 0.0"

zoom	Propagiert den Zoomfaktor (Zahl zw. 2 und 18)

34.9 Operationen ab V 4.11.7

Die folgenden zwei Befehle werden normalerweise zusammen angewendet, Bsp:

```
<Operation ID="PositionObject" opCode="opMapPosition" />
<Operation ID="MoveMode" opCode="opMapMoveMode" forElement="Map"/>

<Menubar> .. <Menu>
 <Item name="mbMoveMode"><Execute name="MoveMode"/></Item>

<DragDrops>
<DragDrop keyState="Ctrl"><Execute name="PositionObject"/></DragDrop>
```

34.9.1 opMapCenterCurrentLocation (ab v5.2.2)

Diese Operation verschiebt das Zentrum der Kartenansicht auf die aktuelle (GPS-)Position. Ein aktivierter GPS-Sensor wird benötigt.

Enthaltene Attribute:

Bezeichnung	Beschreibung
zoom	Definiert den gewünschten Zoomlevel, welcher nach dem Verschieben der Karte eingestellt wird. Wird kein Zoomlevel angegeben, dann wird der Zoomlevel der Karte nicht verändert.

Beispiel

```
<Operation ID="CenterMap"
           opCode="opMapCenterCurrentLocation"
           caption="Karte auf aktuellen Ort zentrieren"
           icon="opLocationOnMap"
           forElement="map"
           zoom="17" />
```

34.9.2 opMapMoveMode

opMapMoveMode ist ein check-Befehl, d.h. das Menu ist angehakt, wenn der Modus aktiv ist. In diesem Modus ist es möglich Komponenten zu verschieben:
Auf selektierter Komponente linke Maus drücken - verschieben - Maus loslassen. In diesem Fall werden keine Bedingungen geprüft und auch keine <AfterPositioning> ausgeführt.

34.9.3 opMapPosition

opMapPosition ist ein Drag&Drop Befehl, mit dem Komponeten positioniert werden können. Bedingungen für den Befehl:

- opMapMoveMode muss aktiv sein.
- Die Komponente muss durch ein `<ContentsIcon>` Element erreicht werden, d.h. die Darstellung ist nur am fehlenden Attribut "Position" gescheitert, und dieses Attribut wird durch diesen Befehl geschrieben. ODER

Es ist eine <PositioningCondition> definiert, dann muss nur diese efüllt sein. Nach dem Positionieren wird <AfterPositioning> ausgeführt.
- Falls ein Condition Element definiert ist, muss dieses auch erfüllt sein.

Falls der aktuelle Zoom-Level der Map die Darstellung verhindern würde, wird diese nach dem Drop automatisch angepasst.

34.9.4 opMapGetCurrentLocation (ab V5.4.2)

Die aktuelle Koordinate kann durch die Variable VAL_CurrentPosition abgefragt werden. Der Befehl ist daher nur in einem Pop-Up Menu sinnvoll.

Beispiel:

```
<Operation ID="MapGetCurrentLocation" opCode="opMapGetCurrentLocation" forElement="map">
  <Navigation> CREATE 1 bisP_Signal MODIFY Position '=VAL_CurrentPosition'
  </Navigation>
</Operation>
```

34.9.5 opMapSetPosition (ab V5.4.2)

Ist eine Drag&Drop Operation, welches auf dem Source-Objekt das Attribut 'Position' setzt.

Beispiel:

```
<Operation ID="MapSetPosition" opCode="opMapSetPosition" forElement="map">
    <Source>CLASS bisP_Signal</Source>
</Operation>
```

34.10 Implementierung StatusMapping

Das Application Element Map implementiert eine Ausgabe auf den Statusbar (vgl. StatusMapping). Folgende IDs sind vorgegeben:

- *MapWGS84* – WGS84 Koordinaten der Mausposition
- *MapCoordCH* – Schweizer Koordinaten der Mausposition
- *MapZoom* – Aktueller Zoomfaktor
- *MapObject* – Object_Display des Objektes unter der Maus
- *MapContents* – Anzahl Objekte, sichtbare, selektierte (wie GridContents)

Beispiel:

```
<StatusMapping>
    <StatusMapEntry ID="MapCoordCH" panel="1"/>
    <StatusMapEntry ID="MapZoom" panel="2"/>
    <StatusMapEntry ID="MapObject" panel="3"/>
</StatusMapping>
```

34.11 Definition von Map - Report ab V 5.1.0

Dieses Kapitel hat nichts mit der Toolkonfiguration zu tun, sondern beschreibt die Definition von Map-Reports, also die Reportdefinition an Objekten der Klasse bisB_MapReport. Damit können Bitmaps aus Karten gemacht werden, die in Reports eingebunden werden können.

Beispiel einer Definition:

```
<Def timeSec="260" width="900" height="600"
     center="7.61 47.5" zoom="16" maxZoom="17">
   <MapService style="BingHybrid" clear="true"/>
   <ContentsShape> [START INSTANCES Building OR START INSTANCES bisAR_OpenArea]
   </ContentsShape>
   <Color default="#FFFF00" alpha="200" mode="solid">
     START "{0C5DE88E-86D4-4CF9-BABA-1E2EEFB3D82B}"
   </Color>
   <ContentsIcon size="16" default="ROOT" minimum="16">START INSTANCES bisP_Signal
   </ContentsIcon>
   <Selection mode="outline" color="#FF0000" alpha="192" width="2" >
      LOOP (VIA Object_To_Parent) CLASS Building
   </Selection>
   <Zoom margin="5">LOOP (VIA Object_To_Parent) CLASS Building
   </Zoom>
</Def>
```

Element `Def` kann folgende Attribute haben:

- `timeSec` Maximale Zeit in Sekunden die für den download der Karten verwendet wird. (default = 120)
- `width / height` Grösse der Bitmap in Pixel.(default = 512) vgl. Auch minZoom maxZoom
- `zoom` wie bei <Map> hat keinen Effekt wenn ein Zoom Element definiert wird (default = 12)
- `center` wie bei <Map> hat keinen Effekt wenn ein Zoom Element definiert wird.
- `minZoom` damit kann ein minimales heranzoomen durch das Zoom Element verlangt warden. Damit ist es möglich, dass das Zoomelement nicht vollständig dargestellt warden. Wenn min Zoom und maxZoom denselben Wert eingestellt haben, dann wird die Bildgrösse (width / height von Def angepasst.
- `maxZoom` damit kann das heranzoomen durch das Zoom Element beschränkt werden. analog gibt es ein minZoom. (vgl. Element Zoom)

Element `MapService` definiert den Hintergrund. Das Element ist Idenisch mit der Beschreibung in <Map>. Hier darf es nur ein solches Element geben.

Element `ContentsShape` definiert die Flächen des Vordergrunds. Das Element ist Identisch mit der Beschreibung in <Map>. Das Eingangsobjekt kommt aus dem Report.

Element `ContentsIcon` definiert die Positionsobjekte im Vordergrund. Das Element ist Identisch mit der Beschreibung in <Map>. Das Eingangsobjekt kommt aus dem Report.

Element `Color` definiert die Färbung der 'ContentsShapes'. Das Element ist Identisch mit der Beschreibung in <Map>. Hier darf es nur ein solches Element geben

Element `Selection` definiert die Objekte, welche selektiert werden. Die Art der Selektion entspricht der Definition des Elmentes in <Map>. Das Eingangsobjekt kommt aus dem Report. Wenn keine Selektion gewünscht ist, dann kann dieses Objekt weggelassen werden.

Element `Zoom` definiert den dargestellten Bereich. Das Eingangsobjekt kommt aus dem Report. Mit dem Attribut `margin` wird der Zusatzrand in Prozent definiert.
Bsp. Mit margin="5" wird das Rechteck am Rand um 5% erweitert, d.h. 10% höher und 10% breiter.

35 FunctionControl (ab v4.11.1)

Das Applikationselement FunctionControl bietet dem Benutzer Funktionen auf Objekten an. Z.B. die Ausgabe eines (angezeigten) Schlüssels. Der Einsatz des FunctionControl ist nur sinnvoll zusammen mit einem weiteren Applikationselement, welches die Objekte darstellt.

35.1.1 Funktionsprinzip

Die Aktivierung einer Funktion des FunctionControls hat das Laden eines zweiten Applikationselements zu Folge. Das zweite Applikationselement ist Typischerweise ein <Form>, wobei für jede Funktion konfiguriert ist, welches Formular angezeigt werden soll.

Benutzeroperationen im zweiten Applikationselement führen durch Propagates von Objekten und Funktionsnamen zur Aktivierung anderer Funktionen oder Funktionsgruppen des FunctionControl. Dies geschieht im Formular durch Skripting mit:

```
FormInformation.PropagateData('FunctionOrGroupName', 0, 0, SelectedObjects);
```

FunctionGroups speichern die Kontextobjekte, auf welche sich die Funktionen beziehen. Diese Kontextobjekte werden an das angeschlossene Applikationselement weitergeleitet.

Im FunctionControl sind die Funktionen in FunctionGroups strukturiert. Das FunctionControl zeigt jeweils die Funktionen einer FunctionGroup an. Die Struktur der FunctionGroups dient der Navigation.

35.2 Beispiele

35.2.1 Mit einem <Form>

Dieses Beispiel verwendet das Applikationselement <Form> mit der ID formFunction um die Objekte darzustellen. Die Funktionen sind in mehrere Funktionsgruppen gegliedert, die folgende statische Hierarchie besitzen:

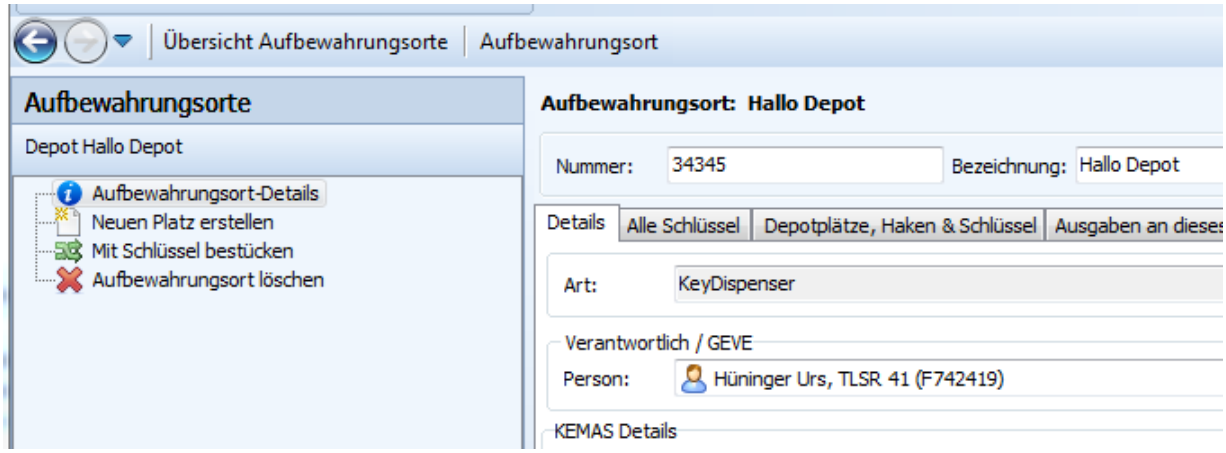
Übersicht Aufbewahrungsorte
- Aufbewahrungsort
  - Steckplatz
  - KEMAS-Modul
  - KEMAS-Depotplan

Die statische Hierarchie wird in der Navigation Area in der rechten Seite angezeigt (welche Funktionsgruppe ist aktiv?).

```
<FunctionControl ID="formFunctions" panel="applicationControl" headerVisible="false"
                 target="formFunction"  >
    <NavigationArea panel="topControl" />
    <FunctionGroup caption="Übersicht Aufbewahrungsorte">
        <Function caption="Aufbewahrungsorte-Liste"  icon="bisKL_Summary"
                  name="LoadUebersicht" formName="Gesamtübersicht Aufbewahrungsorte" />
        <Function caption="Neuen Aufbewahrungsort erstellen" icon="opNew"
                  formName="Neuer Aufbewahrungsort">
            <Modal ok="LoadDepot" />
        </Function>
        <Function caption="Neues KEMAS-Gerät erstellen" icon="opNew"
                  formName="Neuer Aufbewahrungsort (KEMAS)">
            <Modal ok="LoadDepot" />
        </Function>
        <FunctionGroup caption="Aufbewahrungsort" title="$Object_Display_Typed$">
```

```
    <Function caption="Aufbewahrungsort-Details"  icon="bisKL_Summary"
              name="LoadDepot" formName="Übersicht Aufbewahrungsort" />
    <Function caption="KEMAS Depotplan erstellen" icon="opNew"
              formName="Neuer KEMAS Depotplan">
        <Condition>CLASS bisKL_DepotKEMAS -- nur für KEMAS-Depots!</Condition>
        <Modal ok="LoadKEMASDepotplan"/>
    </Function>
    <Function caption="Neuen Platz erstellen"  icon="opNew"
              formName="Neuer Depotplatz">
        <Condition>CLASS ! bisKL_DepotKEMAS  -- nicht für KEMAS-Depots!</Condition>
        <Modal ok="LoadDepotPlatz"/>
    </Function>
    <Function caption="Mit Schlüssel bestücken"  icon="bisKL_ArrowAction"
              formName="Ausgabe auf Aufbewahrungsort" >
        <Modal/>
    </Function>
    <Function caption="Aufbewahrungsort löschen"  icon="opDelete"
              formName="LöschenAllgemein">
        <Condition>
            <!-- keine Schlüssel auf den Plätzen -->
            COUNT (VIA Object_To_Children CLASS bisKL_DepotPlatz
            VIA bisKL_ObjectToKey) = 0
        </Condition>
        <Modal ok="LoadUebersicht"/>
    </Function>

<FunctionGroup caption="Steckplatz" title="$Object_Display_Typed$">
    <ContentsInverse>VIA Object_To_Parent</ContentsInverse>
    <Function caption="Steckplatz-Details"  icon="bisKL_Summary"
              name="LoadDepotPlatz" formName="Übersicht Depotplatz" />
    <Function caption="Steckplatz löschen"  icon="opDelete"
              formName="LöschenAllgemein">
        <Condition>
            CONDITION '=OBJCOUNT("bisKL_ObjectToKey") = 0'
            CONDITION '=OBJCOUNT("bisKL_DepotPlatzToDepotplan") = 0'
            CONDITION '=OBJCOUNT("bisKL_DepotToAuszuloesendeZylinder") = 0'
            CONDITION '=OBJCOUNT("bisKL_DepotToAusloeserZylinder") = 0'
        </Condition>
        <Modal ok="LoadDepot"/>
    </Function>
</FunctionGroup>

<FunctionGroup caption="KEMAS-Modul" title="$Object_Display_Typed$">
    <ContentsInverse>VIA Object_To_Parent</ContentsInverse>
    <Function caption="KEMAS-Modul-Details"  icon="bisKL_Summary"
              name="LoadKEMASModul" formName="Übersicht KEMAS-Modul" />
</FunctionGroup>
<FunctionGroup caption="KEMAS-Depotplan" title="$Object_Display_Typed$">
    <ContentsInverse>VIA Object_To_Parent</ContentsInverse>
        <Function caption="Depotplan-Details"  icon="bisKL_Summary"
                  name="LoadKEMASDepotplan" formName="Übersicht KEMAS-Depotplan" />
        <Function caption="Depotplan löschen"  icon="opDelete"
```

```
                        formName="LöschenAllgemein">
                    <Modal ok="LoadDepot"/>
                </Function>
            </FunctionGroup>

        </FunctionGroup>
    </FunctionGroup>
</FunctionControl>

<Form ID="formFunction" panel="pnMainForm" headerVisible="false"
      ignoreMultiInputObjects="true">
    <FormContents>CLASS bisB_FormObject</FormContents>
    <Propagate operation="load" name="*" target="formFunctions"/>
</Form>
```



Aus den Formularen werden mit der Funktion

```
FormInformation.PropagateData('LoadDepotPlatz', 0, 0, Sender.SelectedObjects);
```

Weitere Funktionen (oder Funktionsgruppen) aktiviert.

35.2.2 Mit einer <Map>

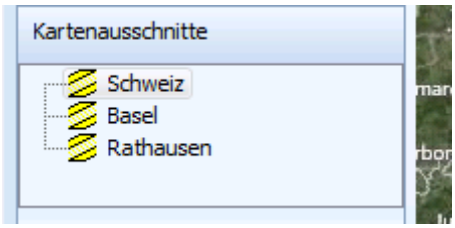
Das Function Control dient dazu, drei vordefinierte Kartenausschnitte anzuzeigen. Das Application Element Map wird dabei durch Propagates gesteuert. Dieselbe Funktionalität liesse sich auch mit einem Toolbar oder einem Menü und opPropagate erreichen.

Die Funktion "CH" zeigt als Kartenausschnitt die ganze Schweiz. Die Funktion stösst das Custom Propagate "CH" an, welches die Map als zoomCenter mit den Parametern der Funktion "CH" erreicht.

Die Funktionen "Basel" und "Rathausen" propagieren Gebäudeobjekte mit "show" an die Map, welche aufgrund der selektierten Gebäude den Kartenausschnitt setzt.

```
<FunctionControl ID="formNav" panel="pnNav" headerVisible="false" >
    <FunctionGroup caption="Kartenausschnitte">
        <Function name="CH" caption="Schweiz" icon="Location" fromParameter="'8 8.4 47.1'"/>
        <Function caption="Basel" icon="Location">
            <Transform>
                START "{6ACEA89E-7B36-48DF-ACE6-CF82670EB525}"
                VIA "Object_To_Children"
            </Transform>
```

```
        </Function>
        <Function caption="Rathausen" icon="Location">
            <Transform>
                START "{6A2EEBC8-BA46-4F25-8BD0-885CAC09CB27}"
                VIA "Object_To_Children"
            </Transform>
        </Function>
    </FunctionGroup>
    <Propagate operation="show" target="Map" />
    <Propagate operation="custom" target="Map" name="CH" nameTarget="zoomCenter" />
</FunctionControl>
```



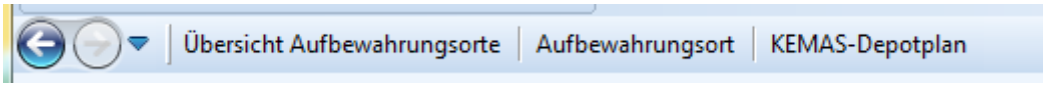
35.3 XML-Attribute

Bezeichnung	Beschreibung
showSelector	`showSelector = <boolean>` Optional. Definiert ob der Funktionsgruppenselektor angezeigt wird. (es muss mindestens eine Funktionsgruppe vorhanden sein) Ab Byron/BIS v5.0.1

target	`target = <text>`
	Applikationselement für welches das FunctionControl ein Load ausführt. Dieses Load wird **vor** den `<Propagate>` ausgeführt.
	Beispiel:
	`<FunctionControl ID="formNav" panel="pnNav" target="formFunctions" >`
	N.B: Bei der Konfiguration muss entweder ein target oder mindestens ein <Propagate> angegeben werden.

35.4 Element Navigation Area

Die Navigation Area ermöglicht das Navigieren in den bereits besuchten Funktionen und Funktionsgruppen bzw. der Liste der Application Work States. Application Work States werden automatisch durch das FunctionControl **nach** dem Propagieren aufgezeichnet. Sie können aber auch mit der Operation opWorkStateRecord oder mit storeWorkStateAfterPropagate aufgezeichnet werden.



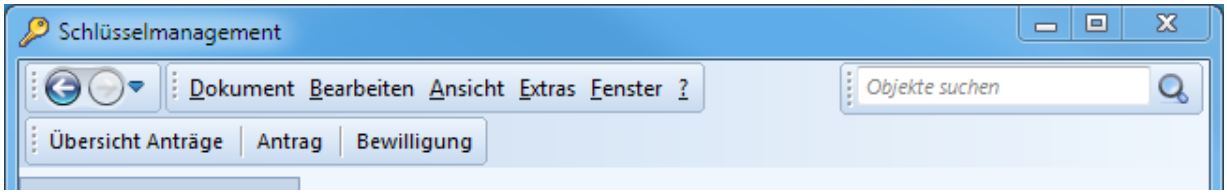
Sie zeigt zweierlei an:

- Links (die beiden Pfeilknöpfe) die Historie der aktivierten Funktionen/application Work States – die Pfeilknöpfe entspechen den Operationen opWorkStateBack und opWorkStateForward.
- Rechts die aktuelle Position in der statischen Funktionsgruppenhierarchie (vgl. erstes Beispiel).

Das Element NavigationArea ist optional. Mit dem Attribut *panel* wird definiert wo (in welchem Panel des Layouts) die Navigation Area dargestellt werden soll.

```
<NavigationArea panel="topControl" />
```

Hinweis: Das Element `NavigationArea` wird ab Byron/BIS v5.0.0 nicht länger benötigt. Der Funktionsgruppenselektor und die Pfeilknöpfe (Vorwärts/Zurück) werden immer als Toolbar angezeigt:



35.5 Element FunctionGroup

Im FunctionControl sind die Funktionen in FunctionGroups strukturiert. Das FunctionControl zeigt jeweils die Funktionen einer FunctionGroup in einem TreeView an. Die Struktur der FunctionGroups dient der Navigation.

Das FunctionControl muss genau eine FunctionGroup enthalten (Standard – FunctionGroup), welche aber wiederum FunctionGroups enthalten darf.

Eine FunctionGroup muss mindestens eine *Function* enthalten.

FunctionGroups speichern die Kontextobjekte, auf welche sich die Funktionen beziehen. Diese Kontextobjekte werden an das angeschlossene Applikationselement weitergeleitet.

Beim Aktivieren einer FunctionGroup wird die erste Function der FunctionGroup aktiviert.

35.5.1 XML-Attribute

Bezeichnung	Beschreibung
caption	Definiert den Text in der Navigation Area. Mehrsprachigkeit: siehe Caption. **N.B**: Darstellungsformeln ($-Notation) werden bis Byron/BIS v4.11.8 hier **nicht** unterstützt. Ab Byron/BIS v5.0.0 können hier auch Darstellungsformeln ($-Notation) verwendet werden. Beispiel: `<FunctionGroup caption="Offerte" …>`
name	Externer Name der FunctionGroup. Eine FunctionGroup (und damit deren Vorgabefunktion) kann über ein Custom Propagate aktiviert werden, indem der Name (und evtl. ein Objekt) an das FunctionControl propagiert wird. Vgl. Propagates
title	Definiert den Text der oberhalb der TreeView dargestellt wird. Ist *title* nicht gesetzt, wird *caption* verwendet. **N.B**: Darstellungsformeln ($-Notation) werden hier unterstützt.

	Beispiel: `<FunctionGroup … title="Offerte $Object_Display_Kontext$">`
icon	`icon = <text>` Optional. Definiert das Icon der Funktionsgruppe im Funktions-gruppenselektor Ab Byron/BIS v5.0.0

35.5.2 Enthaltene Elemente

Function

Eine `FunctionGroup` muss mindestens eine *<Function>* enthalten.

FunctionGroup

Eine `FunctionGroup` kann mehrere `<FunctionGroup>` Elemente enthalten. Die Position der aktuell aktivierten `FunctionGroup` in der statischen Hierarchie wird im Navigation Area angezeigt.

ContentsInverse

Das Element `<ContentsInverse>` enthält eine FilterNavigation, die zur Berechnung der Kontextobjekte verwendet wird, falls eine Funktionsgruppe "von aussen" aktiviert wird. Dies ist zum Beispiel der Fall, wenn eine Funktion aus einer anderen <Application> aufgerufen oder wenn beim Aktivieren einer Funktion eine `FunctionGroup`-Hierarchiestufe übersprungen wird.

N.B.: `<ContentsInverse>` kann weggelassen werden, wenn:

- Die übergeordnete `FunctionGroup` keine Kontextobjekte enthält (Z.b. Übersicht).
- die Funktionsgruppen immer Hierarchiestufe für Hierarchiestufe aktiviert werden (z.B. Übersicht Aufträge - Auftragsdetails - Tätigkeit eines Auftrags).
  Ab Byron/BIS v4.11.8

35.6 Element Function

Die Functions einer aktivierten FunctionGroup werden in einer TreeView dargestellt. Die konfigurierte Hierarchie der Functions entspricht der Hierarchie in der TreeView.

Das Aktivieren einer Function durch den Benutzer geschieht durch Anwählen des TreeView-Knotens.

Das Aktivieren einer Function führt zu einem Propagate an das konfigurierten target bzw. die konfigurierten *<Propagate>*. Folgende Information wird propagiert:

- Kontextobjekte der FunctionGroup (evtl. *<Transform>* geändert).
- Gewünschtes Formular (sofern als *formName* konfiguriert)
- *name* der Funktion
- Code = 0
- Daten aus *fromParameter* (vgl. Attribute)

Im Normalfall können nur diejenigen Functions aktiviert werden, welche die Blätter der Funktionshierarchie bilden. Dieses Verhalten kann ab Byron/BIS v4.11.8 mit dem XML-Attribute `allowActivate` übersteuert werden.

35.6.1 XML-Attribute

Bezeichnung	Beschreibung
allowActivate	`allowActivate = <boolean>` Optional. Erlaubt das Aktivieren einer Funktion (propagate, nicht aber <Modal>) auch wenn die Funktion weitere Funktionen enthält. Vorgabe: `allowActivate="false"` Ab Byron/BIS v4.11.8

caption	`caption = <text>`
	Optional. Definiert den Text der Funktion in der TreeView. Mehrsprachigkeit: siehe Caption
	Ab Byron/BIS v5.6.1 kann eine **Expression** als Caption verwendet werden. Die Expression wird mit den Objekten aus der Contents-Navigation, resp. dem Context-Objekten der Gruppe ausgewertet.
	Beispiel:
	`<Function caption="='xx (' + FORMAT('%d', COUNT) +')'" captionIsBold="=COUNT > 0" >`

captionIsBold	`captionIsBold = <boolean | expression>` Optional. Definiert, ob der Funktionseintrag fett dargestellt wird. Analog `caption` kann hier eine Expression verwendet werden. Beispiel siehe `caption`. Ab Byron/BIS v5.6.1
expanded	`expanded = <boolean | expression>` Optional. Bestimmt, ob der Funktionsknoten initial aufgeklapppt ist. Vorgabe: `expanded="true"` Ab Byron/BIS v5.6.1
allowCollapse	`allowCollapse = <boolean>` Optional. Bestimmt, ob der Funktionsknoten vom Benutzer eingegeklappt werden kann. Vorgabe: `allowCollapse="false"` Ab Byron/BIS v5.6.1
formName	`formName = <text>` Optional. Definiert das Formular, welches beim Aktivieren der Funktion angezeigt werden soll. Das Formular wird in dem durch target definierten Applikationselement angezeigt.

fromParameter	`fromParameter = <expression>`
	Optional. Zusätzlicher Parameter der beim Aktivieren der Funktion propagiert wird (vgl. opPropagate). Bezeichnet der Parameter eine Objektmenge (z.B. "Grid.OBJ_Selection"), dann wird <Transform> auf diese Objekte angewendet (Grid.OBJ_Selection könnte aber auch in <Transform> selbst abgefragt werden.)
	Beispiele:
	`<Function … fromParameter="'8 8.4 47.1'"/>`
	`<Function … fromParameter="18"/>`
	`<Function … fromParameter=" Grid.OBJ_Selection "/>`

icon	`icon = <text>` Optional. Definiert das Icon der Funktion in der TreeView Ab Byron/BIS v4.11.8: dem Icon können mit ‚+' Overlays hinzugefügt werden. Z.B. `icon = "bisP_Order+state_new"`
name	`name = <text>` Optional. Externer Name der Function. Eine Function kann über ein Custom Propagate aktiviert werden, indem der Name (und evtl. ein Objekt) an das FunctionControl propagiert wird. Vgl. Propagates
proceedTo	`name = <text>` Optional. Bezeichnet den Namen einer nächsten Funktion, zu welcher nach der Aktivierung dieser Funktion weitergeleitet wird. Die Weiterleitung hat **keine** weitere Aktivierung zur Folge. Zweck: 1. Funktionen, die Operationen auslösen (Propagate mit `operation="execute"`) können so wieder verlassen werden. 2. Erlaubt das Verwenden einer Funktion, welche weitere Funktionen enthält (sofern `allowActivate="true"`). Erst ab Byron/BIS v4.11.8.

35.6.2 Enthaltene Elemente

Condition

Das optionale Element *Condition* enthält eine Filternavigation, welche aufgrund des Kontextobjekts der Funktionsgruppe bestimmt, ob die Funktion sichtbar ist, bzw. aktiviert werden kann.

Contents

Das optionale Element *Contents* enthält eine Filternavigation, welche aufgrund des Kontextobjekts der Funktionsgruppe bestimmt, welche Objekte für die `caption` und `captionIsBold`-Expressions verwendet werden.

Erst ab Byron/BS v5.6.1.

Function

Mit dem Element *Function* werden enthaltene Funktionen konfiguriert. Enthält eine Function weitere Funktionen, dann kann sie nicht aktiviert werden. Nur die Eigenschaften *caption* und *icon* haben noch eine Bedeutung.

Modal

Das Element *Modal* zeigt an, dass das Formular in einem separaten Fenster modal geöffnet wird. Das Element kann das Attribut *ok* enthalten, welches angibt, welche Funktion aufgerufen/aktiviert wird, wenn das Formular mit "OK" beendet wird.

```
<Function caption="Depotplan löschen" name="deleteDepot" icon="opDelete" formName="LöschenAllgemein">
    <Modal ok="LoadDepot"/>
</Function>
```

Achtung: Wird mit einem modalen Formular ein neues Objekt erzeugt, dann muss das Formular entsprechend programmiert werden, da es im Modus *Öffnen* geöffnet wird. Beispiel:

```
<Function caption="Mitarbeiter anlegen" name="newMitarbeiter" icon="Person+state_new"
          formName="PersonAnlegen">
    <Modal ok="Personenliste"/>
</Function>
```

Formularskripting (Reihenfolge der Operationen siehe Formularreferenz):

```
procedure BisDataSourceOperate(theOp: TBisDataOperation; obj: TDbBaseObject);
begin
  if theOp = bdoLoadData then begin
    BisDataSource.ObjectType := 'Person';
    BisDataSource.Operate(bdoPrepareCreate);
    BisDataSource.Operate(bdoSetDefault);
    BisDataSource.Container := FormInformation.ExternalSelection; // hier anpassen!
  end;
end;
```

Erst ab Byron/BIS v5.0.1:

Der Name der modalen Function wird als Parameter an die FormInformation des Formulars gesendet. Um diese zu verwenden, muss an der FormInformation das Ereignis `OnSetParameter` verwendet werden. Der Parameter wird als String in der Form `VAL_CurrentFunction=NameOfFunction` übermittelt.

Erst ab Byron/BIS v5.0.1:

Nach dem Schlessen des Formulars mit „OK"wird ein Propagate mit den durch das Formular zurückgegebenen Objekten ausgeführt (<Transform> wird dabei ignoriert). Danach gibt es folgende Möglichkeiten:

1. es wurde ein ok="<FunctionName>" konfiguriert.
   Die angegebene Funktion wird mit den Objekten aufgerufen, welche das Formular zurückgegeben hat.
2. die CurrentFunction wurde durch das Propagate verändert
   (z.B. wurde ein Load an das FunctionControl gesendet)
   Es werden keine weiteren Aktionen durchgeführt
3. sonst
   Die Funktion, welche vor dem Aufruf des modalen Formulars aktiv war, wird erneut aktiviert

Transform

Das Element *Transform* enthält eine Filternavigation, welche beim Aktivieren der Funktion entweder auf die Kontextobjekte oder auf die durch *fromParameter* gelieferten Objekte angewendet wird,

35.7 Propagates

Das FunctionControl steuert die anderen ApplicationElemente mit Propagates (Load oder Custom). Kommt es zu Propagate-Zyklen, dann kann ab v4.11.8 die Methode zur Erkennung der Zyklen geändert werden. Vgl. Application (classicPropagation).

35.7.1 Propagates (Input)

Das FunctionControl wertet sowohl load als auch custom propagates aus wie folgt:

- Der Parameter *name* der additional Information wird als Bezeichnung (*name*) der zu aktivierende Function oder FunctionGroup interpretiert.
- Die übergebenen Objekte werden zu Kontextobjekten der aktivierten FunctionGroup.

Erst ab Byron/BIS v5.0.1:

Wird das Custom-Progagate `ReloadFunctions` empfangen, werden alle Functions der aktuellen Funktionsgruppe, im Speziellen die Conditions, neu ausgewertet und entsprechend dargestellt, bzw. ausgeblendet.

Ab Byron/BIS v5.2.1 wertet das FunctionControl Show-Propagates (bzw. Show-Operationen über Drag-Drop) aus wie folgt:

- Ist kein Propagate konfiguriert, welches auf den Namen `"dispatchShow"` matched und ungleich «*» ist, dann passiert gar nichts.
- Ist mindestens ein Propagate konfiguriert, welches auf den Namen `"dispatchShow"` matched und ungleich «*» ist, dann wird ein Propagate mit dem Namen `"dispatchShow"` ausgelöst. Dieses wird auch weitergeleitet, wenn der Name des Propagates «*» ist.

Es ist zu beachten:

- Ein Show (Propagate oder Drop auf die Komponente) ändert a priori nichts am Zustand des Function-Controls (aktuelle Funktion, gespeicherte Objekte).
- Da das FunctionControl die Show-Operation nur weiterpropagieren kann, wird auch **keine Meldung angezeigt**, wenn das Objekt nicht gefunden wurde. Dies ist vor allem zusammen mit dem nächsten Punkt ungünstig.
- Ein Show kann auch durch Zeige-Operationen „angeschlossener" Elemente ausgelöst werden. Siehe auch Grid (dropShowTryOthersFirst). Dem aufrufenden Element wird dabei immer signalisiert, dass das Objekt gefunden wurde (auch hier keine Meldung).
- Die Regeln, wann ein Propagate mit `dispatchShow` matched, wurden in Byron/BIS v5.2.2 angepasst. Ein einzelnes «*» im Namen des Propagates wird in diesem speziellen Fall ignoriert.

35.7.2 Propagates (Output)

Folgende Propagates werden vom Applikationselement beim Aktivieren einer Funktion ausgelöst:

- Ist ein target konfiguriert, dann wird ein "load" an diese Applikationselement geschickt. Ist formName konfiguriert, wird zusätzlich zu den Kontextobjekten der FunctionGroup das gewünschte Formular(objekt) propagiert. Die additional information enthält die Bezeichnung der Funktion, sowie weite Parameter (vgl. Function).
- Die konfigurierten <Propagates> werden ausgelöst. Die additional information enthält die Bezeichnung der Funktion, sowie weite Parameter (vgl. Function). Beispiel: siehe hier.

35.8 Parameter für die FilterNavigation

Siehe auch: Parameter für die FilterNavigation

Parameter	Datentyp	Bemerkung
VAL_CurrentFunction	string	Name der aktuell aktiven Funktion
VAL_CurrentFunction-Group	string	Name der aktuell aktiven FunktionGroup

Der Parameter muss die Element-ID als Prefix beinhalten.

```
<Condition> BLOCK "=VAR('FormFunctions.VAL_CurrentFunction')='ld_OffertDetail'"</Condition>
```

36 IndoorViewer BETA ab v5.2.1

Bindet einen Web-Browser mit dem IndoorViewer von NavVis ein. Mit Hilfe des IndoorViewers kann, in 3D, durch Räumlichkeiten navigiert werden (ähnlich zu Google StreetView).

Das Applikationselement kann Propagates empfangen, um zu bestimmten Koordinaten zu springen und auch Propagates aussenden um POIs (Points of Interest, Positionskomponenten) auszuwählen.

36.1 Beispiel

```
<IndoorViewer ID="ivRaumbuch" panel="pnFormIndoorViewer" headerVisible="false">
  <Contents>
    [ OR LOOP (VIA Object_To_Parent) ] CLASS Floor
  </Contents>
  <Url>https://de.navvis.com/BI-office</Url>
  <Propagate operation="load" target="form"/>
</IndoorViewer>
```

36.2 Attribute

Bezeichnung	Beschreibung

36.3 Elemente

Bezeichnung	Beschreibung
Contents	FilterNavigation zu angezeigtem Indoor-Bereich. **BETA:** Später wird hier evtl. die (Teil-)URL für die Anzeige hinterlegt. Aktuell kann damit gesteuert werden, ob ein Objekt angezeigt wird oder nicht (z.B. BLOCK bei allen Stockwerken, welche nicht dem Byron-Hauptsitz entsprechen).
Url	URL der NavVis-IndoorViewer-Komponente. **BETA:** Später wird diese, oder Teile der URL evtl. aus dem Contents-Objekt gelesen.

36.4 Empfangene Propagates

Bezeichnung	Beschreibung
show	Ein Objekt mit den Attributen `bisB_WGS84Coordinate_X` und `bisB_WGS84Coordinate_Y` (WGS84-Koordinaten) welche verwendet werden um zu einem bestimmten Punkt zu springen. `<Propagate operation="show" target="indoorViewer">`

36.5 Ausgelöste Propagates

Bezeichnung	Beschreibung
	Beim Auswählen von POIs (Points of Interest, Positionskomponenten) werden die konfigurierten Propagates ausgelöst.

	Als Objekte werden, anhand der im IndoorViewer von NavVis vergebenen ID, die zugehörigen Byron/BIS-Objekte übergeben (mit Hilfe des Attributs `bisB_IndoorViewerId`). Hierzu wird folgende FilterNavigation verwendet: `START VALUE bisB_IndoorViewerId = <POI-ID>`

37 Operation

Eine Operation definiert die Aktion, welche durch Menu und Drag&Drop ausgelöst werden.

Alle Operationen stehen innerhalb des Operations Element.

37.1 XML-Attribute

Bezeichnung	Beschreibung
ID	`ID = <Text>` Namen für die Operation. Diese wird für das Element Execute verwendet, um die Operation zu referenzieren. Kein Default, die ID muss zwingend gesetzt sein.
opCode	`opCode = <Text>` definiert die Operation, welche ausgeführt wird. Siehe Tabelle Liste der Operation (opCode)
caption	`caption = <Text>` Bezeichnung für die Operation. Diese wird für Menüeinträge in Item und DragDrop verwendet. Optional default ev. von der Operation vorgegeben
shortcut	`shortcut = <Text>` Tastenkombination für die Operation. Diese wird für Menüeinträge in Item verwendet. Optional, default = siehe Liste
hint	`hint = <Text>` Hint für die Operation. Diese wird für ToolbarButton verwendet. Optional, default = leer
icon	`icon = <Text>` Icon für die Operation. Dieses wird für Toolbar-Button verwendet. Optional, default = leer Ab Byron/BIS v4.11.8: dem Icon können mit ‚+' Overlays hinzugefügt werden. Z.B. `icon = "bisP_Order+state_new"`
onElement	`onElement = <Text>` Der Wert referenziert ein Application Element. Damit ist die Operation nur auf diesem Appllication-Element zulässig. Wenn der Fokus auf einem anderen Application Element liegt, dann ist die Operation nicht möglich (z.B. Menu disabled). Die Angabe ist optional. Wenn das Attribut fehlt, dann ist die Operation für alle Application Element möglich. (vgl. forElement)
forElement	`forElement = <Text>` Der Wert referenziert ein Application Element. Damit wird die Operation immer auf diesem Element ausgeführt, unabhängig vom Fokus (vgl. onElement). Wenn für eine Operation nur ein bestimmeter Typ Application-Element unterstützt wird, dann nennt sich dieses Attribut auch dem entsprechend. zB. forGrid, forTree etc. dies ist dann jeweils bei der Operation vermerkt.
ignoreImplicitGroupRights	`ignoreImplicitGroupRights = <Boolean>` Optional. Gibt an, ob implizite Gruppenrechte für diese Operation (z. Bsp. "Objekte mit enthaltenen Objekten löschen" für die Operation `opDelete`) ignoriert, dh. nicht verlangt, werden sollen. Default: `ignoreImplicitGroupRights="false"` Erst ab Byron/BIS v4.11.3

37.2 XML-Elemente

Bezeichnung	Beschreibung
Source	Filternavigation für die Quell-Objekte
Target	Filternavigation für die Ziel-Objekte
Condition	Mit der Condition-Filternavigation kann das Ausführen der Operation verhindert werden. Wenn kein Condition-Element existiert, wird die Operation immer ausgeführt. Die Eingangsobjekte (Input) hängen von der Operation ab. Beim Drag&Drop werden die Source-Objekte verwendet. **Achtung**: folgende Operations unterstützen <Condition> **nicht**: • opActualizeElement • opCopyTemplate • opFormEdit • opNavParameter • opUndo • opUserSettings • opUserSettingsCopyFrom • opUserSettingsCopyTo • opUserPassword • opWorkStateBack • opWorkStateForward • opWorkStateRecord
Confirmation	Ist für Confirmation ein Text angegeben, wird vor der Ausführung der Operation ein Bestätigungsdialog mit diesem Text angezeigt. Die Operation wird nur ausgeführt, wenn der Benutzer "Ja" wählt. Hinweis: Die impliziten Bestätigungen von `opDelete` und `opXXXBind` können nicht verändert werden. Erst ab Byron/BIS v4.11.3 Ab Byron/BIS v5.0.0 kann die $-Notation verwendet werden. Für die Auswertung wird das erste Source-Objekt verwendet. Die Auswertung wird nur von den Operationen unterstützt, welche auch `Condition` unterstützen!

37.3 Vordefinierte Parameter für die Condition-FilterNavigation:

Parameter	Datentyp	Bemerkung
VAL_TargetElementId	Text	Die ID des Ziel-Elements beim Drag&Drop
OBJ_Source	Objekt	Quell-Objekte beim Drag&Drop. Ab Byron/BIS v 5.3.2 kann OBJ_Source einer anderen Operation abgefragt werden mit `<Sequence-ID>.OBJ_Source` z.B. Kopieren.OBJ_Source.

		N.B: OBJ_Source ist nur gesetzt, während die Operation ausgeführt wird.
OBJ_Target	Objekt	Ziel-Objekte beim Drag&Drop Ab Byron/BIS v 5.3.2 kann OBJ_Target einer anderen Operation abgefragt werden mit `<Sequence-ID>.OBJ_Target` z.B. Kopieren.OBJ_Target. **N.B**: OBJ_Target ist nur gesetzt, während die Operation ausgeführt wird.

Beispiel:

```
<Operations>
    <Operation ID="SelectAll" opCode="opSelectAll"
               onElement="grid"shortcut="Ctrl+A"/>
    <Operation ID="viewEdit" opCode="opEditGridView" forGrid="grid/>
    <Operation ID="viewNew" opCode="opNewGridView" forGrid="grid"
</Operations>
```

37.4 Liste der Operation (opCode)

opCode	Bezeichnung
opActualize	Aktualisieren
opActualizeAll	Alles aktualisieren
opActualizeElement	Element aktualisieren
opAddToClipboard	Im Clipboard ablegen
opBind	Assoziation erzeugen
opCopy	Kopieren
opCopyAttrValues	Attributwerte kopieren
opCopyTemplate	Kopieren
opCreate	Erzeugen
opDelete	Löschen
opDeleteAttribute	Attribut löschen
opDeleteGridView	Ansicht löschen
opEditAttribute	
opEditGridView	Ansicht bearbeiten
opExecuteAction	MenüAktion ausführen
opExecuteFilterNavigation	FilterNavigation ausführen
opExport	Exportieren
opFilesSendTo	Senden an
opFormEdit	Formular bearbeiten

opCode	Bezeichnung
opFormEditText	Formular als Text
opGridOptimizeRowHeight	Optimierte Zeilenhöhe
opGridTrackScroll	Aktiven Bildlauf
opImport	Importieren
opLinkFiles	Dokument hier erzeugen
opMove	Verschieben
opNavParameter	Parameter
opNavSubmenu	Objekte
opNewGridView	Ansicht neu
opObjectRightsEdit	Objektrechte bearbeiten
opOpen	Öffnen
opOpenContainer	im Baum öffnen
opPaste	Einfügen
opPrint	Seitenansucht/Drucken
grPrinter	Grafik drucken
opPropagate	
opPropagateSubmenu	
opProperties	Eigenschaftsblatt
opResetAttribute	Attribut auf Vorgabe
opSearch	Suche öffnen
opSelectAll	Alle markieren
opSequence	
opShellExecute	
opShortCutCreate	Verknüpfung erstellen
opShortcutMove	Verknüpfung verschieben
opShortcutRemove	Verknüpfung entfernen
opShow	Zeigen
opShowHistory	Objekt-Historie anzeigen
opShowPage	Seite aus Pages-Applikationselement anzeigen
opSwitchMinimized	Element minimieren / maximieren
opTreeGotoParent	Übergeordnetem Element
opTreeHistoryBackward	"Zurück" im Applikationselement Tree
opTreeHistoryForward	"Vorwärts" im Applikationselement Tree

opCode	Bezeichnung
opUnbind	Assoziation löschen
opUndo	Rückgängig
opUserPassword	Kennwort ändern
opUserSettings	Benutzereinstellungen
opUserSettingsCopyFrom	Verknüpfungen kopieren von
opUserSettingsCopyTo	Verknüpfungen kopieren zu
opVisibleOnOff	Element ein-/ausblenden
opWorkStateBack	"Zurück" in der Anwendungshistorie
opWorkStateForward	"Vorwärts" in der Anwendungshistorie
opWorkStateRecord	Aufnehmen eines Standes der Anwendungshistorie
opWorkStateReload	

Vergleiche L:\Dokumentation\Technische Dokumentation\Übersicht über die Operationen.xls

37.5 opActualize

Die Operation Actualize entspricht der Operation "Aktualisieren" (F5).

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="Actualize" opCode="opActualize"/>
```

37.6 opActualizeAll

Die Operation ActualizeAll entspricht der Operation "Alles aktualisieren" (SHIFT + F5).

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="ActualizeAll" opCode="opActualizeAll"/>
```

37.7 opActualizeElement (erst ab Byron/BIS v4.8.6)

Die Operation ActualizeElement entspricht der Operation "Aktualisieren", wobei aber nur das angegebene Element aktualisiert wird.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden. Sie unterstützt <Condition> **nicht**.

Beispiel

```
<Operation ID="ActualizeTree" opCode="opActualizeElement" forElement="tree"/>
```

37.8 opAddToClipboard

Die Operation AddToClipboard kopiert das selektierte Objekt des Application Elements in die Zwischenablage. Mit dem Attribut `cutFormat` kann gesteuert werden, ob das Objekt beim Einfügen aus der Zwischenablage kopiert (false) oder ausgeschnitten/verschoben (true) werden soll.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel 1 (Kopieren)

```
<Operation ID="AddClipboard" opCode="opAddToClipboard"/>
```

Beispiel 2 (Ausschneiden)

```
<Operation ID="CutClipboard" cutFormat="true" opCode="opAddToClipboard"/>
```

37.9 opBind

Die Operation Bind verknüpft die Source-Objekte mit den Target-Objekten für die angegebene Assoziation.

Die Operation kann in Menüs, PopupMenüs und im Drag&Drop verwendet werden.

Das `rebind`–Attribut gibt an, ob ein Bind oder Rebind ausgeführt wird:

Wenn `rebind="false"` ist, wird ein Bind ausgeführt: `Source.Bind(assoc, Target)`
wenn `rebind="true"` ist, wird ein Rebind ausgeführt: `Source.Rebind(assoc, Target)`

Das `direction`–Attribut gibt die Richtung für ein Bind oder Rebind an:

Wenn `direction="source-target"` ist (Default), wird ein Bind oder Rebind von Source zum Target ausgeführt: `Source.Bind(assoc, Target) oder Source.Rebind(assoc, Target)`
wenn `direction="target-source"` ist, wird ein Bind oder Rebind vom Target zur Source ausgeführt: `Target.Bind(assoc, Source) oder Target.Rebind(assoc, Source)`

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements oder die Drag-Objekte verwendet.

Als Eingangsobjekte (Input) für die `Target`-Filternavigation werden die Objekte in der Zwischenablage oder das Drop-Objekt verwendet.

Beispiel 1

Die folgende Operation führt ein Bind von Source zum Target aus:

```
<Operation ID="AddToMyList" opCode="opBind" rebind="false"
           assoc="bisB_ObjectsToList" onElement="myList">
    <Source>CLASS Root</Source>
    <Target>START USER VIA bisB_UserToList</Target>
</Operation>
```

Beispiel 2

Die folgende Operation führt ein Rebind vom Target zur Source aus:

```
<Operation ID="PersonZuordnen" opCode="opBind" rebind="true"
           direction="target-source" assoc="Room_Of_Component"
           caption="Raum zuordnen (Person --> Raum)">
    <Source>CLASS Space</Source>
    <Target>CLASS Person</Target>
</Operation>
```

37.9.1 Hinweise

- Für zu-1-Assoziationen (n:1 laut Info-Spalte im Modell) muss des rebind-Attribut auf "true" sein, da sonst eine Fehlermeldung bezüglich der Kardinalität erscheint!

- Für zu-n-Assoziationen (1:n oder n:n laut Info-Spalte im Modell) muss des rebind-Attribut auf "false" sein, da sonst zuerst alle anderen Objekte von der Assoziation entfernt werden!

37.10 opCopy

Die Operation Copy kopiert die Source-Objekte unter das Target-Objekt des Application Elements.

Die Operation kann in Menüs, PopupMenüs und im Drag&Drop verwendet werden.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation werden die Objekte in der Zwischenablage oder die Drag-Objekte verwendet.

Als Eingangsobjekte (Input) für die `Target`-Filternavigation wird die Selektion des Application Elements oder das Drop-Objekt verwendet.

Beispiel

```
<Operation ID="CopyMenu" opCode="opCopy">
    <Source>CLASS "bisB_MenuItem"</Source>
    <Target>[CLASS "bisB_MenuRoot" OR CLASS "bisB_MenuItem"]</Target>
</Operation>
```

37.11 opCopyAttrValues

Die Operation CopyAttrValues ruft den Dialog "Attributewerte kopieren" auf.

Beispiel

```
<Operation ID="actCopyAttrValues" opCode="opCopyAttrValues"/>
```

37.12 opCopyTemplate

Die Operation `CopyTemplate` kopiert die `Source`-Objekte unter das `Target`-Objekt des Application Elements. Die Objekte in `MandatorySource` werden auch kopiert, jedoch unter das Objekt, das über die `Container`-FilterNavigation erreicht wird.

Der Input der `Container`-FilterNavigation ist das einzelne `MandatorySource`-Objekt. Wird die `Container`-FilterNavigation leer gelassen, ist der Default `VIA "Object_To_Parent"`.

Die Operation führt ein Deepcopy aus. Für jedes Objekt in `Source`- und `Mandatory`-Source werden die Children einzeln durch die `IncludeChildCondition`-FilterNavigation gefiltert. Falls die resultierende Objektmenge = 0 ist, wird das Child nicht kopiert. Mit `BLOCK` kann so ein Deepcopy verhindert werden.

Die in `InnerAssociations` angegebenen Assoziationen werden unabhängig vom Copy-Flag an der Objektklasse mitkopiert. Ist ein Objekt über eine `InnerAssociation` mit einem in dieser Operation kopierten Objekt verknüpft, so wird das originale Objekt, durch das neu erstellte ersetzt, ansonsten bleibt das originale Objekt über die Assoziation verknüpft(!!!). Um eine innere Assoziation so zu konfigurieren, dass sie **entweder** mit einem **kopierten Objekt oder** aber **gar nicht** verknüpft wird, kann das Attribut `onlyBindIfTargetCopied` auf `true` gesetzt werden. Der Default dieses Attributs ist `false`.

Der Wert von `ClassAttribute` enthält den Namen des Attributs, welches den Namen der zu erstellenden Klasse beinhaltet. So kann ein Vorlagenobjekt, welches von einer Vorlagen-Objektklasse abgeleitet ist kopiert werden und gleichzeitig ein Klassenwechsel stattfinden. Ist der Parameter `ClassAttribute` leer, wird automatisch das Attribut `Object_TypeId` verwendet. Falls der Wert des in `ClassAttribute` angegebenen Attributs leer ist, wird das Objekt nicht kopiert.

Falls die Assoziation in `TemplateAssociation` am neuen Objekt erlaubt ist, wird die Vorlage, aus der das neue Objekt hervorgeht, an diese Assoziation gebunden. Dieses Element ist Optional.

Das Element `RenameObjects` bietet die Möglichkeit, kopierte Objekte umzubenennen. `Filter` bestimmt, ausgehend vom **Parent der Selektion oder** bei Drag&Drop vom **DropTarget** (nicht der Source-Navigation!), welche Kopien welcher Objekte umbenannt werden müssen. Dazu wird `formula` (des `Renaming`-Elements) auf dem original-Objekt ausgewertet und in das Attribut `intoAttribute` geschrieben. Im Element `Renaming` kann eine Filternavigation angegeben werden, innerhalb welcher Objektmenge ein Attribut eindeutig sein muss (ausgehend vom Kopie-Objekt). Ist das Attribut dann nicht eindeutig, wird `$counter$` erhöht. Der Text `$counter$` in `formula` wird durch "(n)" ersetzt sofern er grösser ist wie 1, ansonsten wird `$counter$` aus dem Text entfernt. Siehe dazu Beispiel 2. `formula` kann auch ein Text aus dem Textkatalog sein, welcher wiederum eine Formel enthält. Falls der `$counter$` nicht hochgezählt werden soll, sollte die Navigation innerhalb des `Renaming`-Elements auf `BLOCK` gesetzt werden.

ACHTUNG:

- Diese Operation löst ein OnCopy-Event aus!
- Diese Operation unterstützt <Condition> **nicht**.

Das Beispiel 1 kopiert Räume und ihre Komponenten (bzw. Devices). Die Räume werden unter dem `Target` eingefügt, die Komponenten jedoch unter dem Parent der zu kopierenden Komponente. Die kopierten Komponenten werden dann mit dem neuen, kopierten Raum über Room_Of_Component verknüpft. Die einzelnen Komponenten sind zu ihren original Komponenten über bisB_DeviceToTemplate verknüpft.

37.12.1 BEISPIEL 1

```
<Operation ID="DeepCopyOperation" opCode="opCopyTemplate" UseObjectRights="true">
    <Source>
        PASS
    </Source>
    <MandatorySource>
        [CLASS "Room" ]
        [ OR LOOP(VIA "Object_To_Children") CLASS "Space"]
        VIA "Room_To_Component" CLASS "Device"
    </MandatorySource>
    <Container>VIA "Object_To_Parent"</Container>
    <IncludeChildCondition>CLASS "Space"</IncludeChildCondition>
    <Target>
        [ CLASS "Floor" OR CLASS "Room" OR CLASS "MV_Zone" ]
    </Target>
    <InnerAssociations>
        <InnerAssociation onlyBindIfTargetCopied="true">Room_Of_Component</InnerAssociation>
    </InnerAssociations>
    <ClassAttribute>Object_TypeId</ClassAttribute>
    <TemplateAssociation>bisB_DeviceToTemplate</TemplateAssociation>
    <RenameObjects>
        <Filter>
            CLASS "Space"
        </Filter>
        <Renaming formula="$name$ - Kopie $counter$" intoAttribute="Name">
            VIA "Object_To_Children"
            CLASS "Space"
        </Renaming>
    </RenameObjects>
</Operation>
```

37.12.2 BEISPIEL 2

```
<Operation ID="DeepCopyOperation" opCode="opCopyTemplate" UseObjectRights="true">
    <Source>
        PASS
    </Source>
    <MandatorySource>
        CLASS "bisB_PersonContainer"
    </MandatorySource>
    <Container>VIA "Object_To_Parent"</Container>
    <IncludeChildCondition>CLASS "Person"</IncludeChildCondition>
    <Target>
        [ CLASS "bisB_PersonContainer" ]
    </Target>
    <ClassAttribute>Object_TypeId</ClassAttribute>
    <RenameObjects>
        <Filter>
            CLASS "bisB_PersonContainer"
        </Filter>
        <Renaming formula="$name$ - Kopie $counter$" intoAttribute="Name">
            VIA "Object_To_Children"
        </Renaming>
    </RenameObjects>
</Operation>
```

37.13 opCreate

Die Operation Create erzeugt Objekte von der angegebenen Objektklasse.

Mit dem Attribut `recursive="false"` kann verhindert werden, dass auch abgeleitete Klassen eingefügt werden können. Default = "`true`".

Mit dem Attribut `intoElement=`<FormID> kann verhindert werden, dass ein Fenster für das neue Objekt geöffnet wird. Stattdessen wird im Appl. Element mit der ID=<FormID> das neue Objekt angezeigt. Dieses Appl. Eelement sollte natürlich ein <Form> sein.

Einfaches Beispiel

```
<Operation ID="CreateContainer" opCode="opCreate" class="bisKL_Container"/>
```

Wird keine FilterNavigation für Container, bzw. Association angegeben, wird das Objekt "global" erzeugt.

Der Container für das neue Objekt wird aus der (optionalen) Container-Filternavigation bestimmt.

Als Eingangsobjekte (Input) für die `Container`-Filternavigation wird der Container des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel (abgeleiteten Klassen erlaubt)

```
<Operation ID="CreateMenuItem" opCode="opCreate" class="bisB_MenuItem">
    <Container>[CLASS "bisB_MenuRoot" OR CLASS "bisB_MenuItem"]</Container>
</Operation>
```

Beispiel (keine abgeleitete Klassen erlaubt)

```
<Operation ID="CreateContainer" opCode="opCreate" class="bisKL_Container"
           recursive="false">
    <Container>CLASS "bisKL_Container"</Container>
</Operation>
```

Für das referenzierte Erzeugen muss statt dem `Container`-Element ein `Association`–Element und der Name der Assoziation angegeben werden. Als Container für das neue Objekt wird der Defaultcontainer laut Modell verwendet.

Als Eingangsobjekte (Input) für die `Association`-Filternavigation wird die Selektion des Application Elements verwendet.

Beispiel

```
<Operation ID="CreateRoomReservationSingle" opCode="opCreate"
           class="bisR_RoomReservationSingle">
    <Association name="bisT_TerminToResource">CLASS Room</Association>
</Operation>
```

Wenn das Attribut `intoElement` gesetzt ist, wird das neue Objekt im angegebenen Applikationselement erzeugt. So kann verhindert werden, dass beim Einfügen eines neuen Objektes ein neues Formular erscheint.

Beispiel

```
<Operation ID="CreateMenuItem" opCode="opCreate" class="bisB_MenuItem" intoElement="form">
    <Container>[CLASS "bisB_MenuRoot" OR CLASS "bisB_MenuItem"]</Container>
</Operation>
```

Modales Erzeugen (erst ab v5.4.5): Wenn das Attribut `modal="true"` gesetzt ist, wird das Erzeugen des Objekts in einem modalen Fenster durchgeführt. Die Angabe von intoElement hat eine höhere Priorität als modal.

37.14 opDelete

Die Operation Delete löscht die Source-Objekte.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="Delete" opCode="opDelete">
    <!-- keine Root-Objekte -->
    <Source>COUNT (VIA Object_To_Parent) > 0</Source>
</Operation>
```

37.15 opDeleteAttribute

Die Operation DeleteAttribute löscht den Wert für das aktuelle Attribut des Application Elements für alle selektierten Objekte.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden und für Applikationselemente, welche ‚Attributselektion' unterstützen, zB. Grid.

Beispiel

```
<Operation ID="DeleteAttribute" opCode="opDeleteAttribute"/>
```

Mit name kann ein Atrribut für die Operation definiert werden, damit funktioniert die Operation für alle Appplikationselemente.

Beispiel mit Name

```
<Operation ID="DeleteAreaGraphic" opCode="opDeleteAttribute" name="bisB_AreaGeometrie"/>
```

37.16 opEditAttribute

Die Operation EditAttribute erlaubt das Editieren des aktuellen Attributs des Application Elements für alle selektierten Objekte.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="EditAttribute" opCode="opEditAttribute"/>
```

37.17 opExecuteAction

Die Operation ExecuteAction führt eine Menü-Aktion mit der angegebenen Aktions-ID aus.

Beispiel

```
<Operation ID="LinkExtDoc" opCode="opExecuteAction" actionID="LinkExtDoc"/>
```

37.18 opExecuteFilterNavigation (ab v4.11.3)

Die Operation ExecuteFilterNavigation führt eine FilterNavigation (wenn nötig mit Schreibtransaktion) auf den selektierten Objekten aus. Die Operation kann in Menüpunkten und als Drag&Drop-Operation verwendet werden.

```
<Operation ID="ExecuteMyFilterNav" opCode="opExecuteFilterNavigation">
    <Source>
        -- wird als Input in Execute gegeben
    </Source>
    <Target>
        -- Für Drag&Drop Operations
    </Target>
    <Condition>
      -- Für Drag & Drop: überprüfen, dass die Source eine Person und Target ein Raum
      [CLASS "Person" PICK 1 OR RECALL "OBJ_Target" CLASS "Space" PICK 1]
      BLOCK "=COUNT < 2" – Condition false, wenn nicht mindestens zwei Objekte
    </Condition>
    <Execute>
        SAVEAS input
        [
            -- Falls Drag&Drop haben wir ein Target, sonst nicht
            RECALL "OBJ_Target"
        ELSIF = 0
            -- Dann nehmen wir z.B. die Tree-Selektion
            RECALL "tree.OBJ_Selection"
        ]
        SAVEAS myVar "=ATTRIBUTE('Name')"
        RECALL input
        MODIFY Name "=myVar + '_' + ATTRIBUTE('Name')"
    </Execute>
</Operation>
```

37.19 opExport

Die Operation Export ruft den Dialog zum Exportieren der selektierten Objekte als Text, CSV, SYM oder BIX auf.

Attribute:

supportSYM Ermöglicht den Export von Condor-Symboldateien (*.sym), default = false

Beispiel:

```
<Operation ID="Export" opCode="opExport" supportSYM="true"/>
```

Export von Grafik in diverse Formate ist möglich, wenn das Element ein Grafikelement ist (Graphic oder VdrawGraphic). Wenn der Export nur Grafik unterstützen soll, dann kann dies durch eine Elementzuordnung erreicht werden:

Beispiel:

```
<Operation ID="ExportGraphic" opCode="opExport" forElement="graphic"/>
```

Wenn der Export universal auf dem Fokuselement erfolgt, dann kann mit dem Attribut style noch angegeben werden, was bevorzugt werden soll:

```
style= graphicPreferred, graphicOnly, graphicProhibited (default = graphicPreferred)
```

graphicPreferred: Bei einem Grafikelement wird der Grafikexport verwendet.

graphicOnly: Es wird nur Grafikexport, (Grafikelemente) unterstützt.

graphicProhibited: Es wird nie ein Grafikexport gemacht, d.h. auf Grafikelmente kann z.B. ein Bis-Export gemacht warden.

Beispiel:

```
<Operation ID="Export" opCode="opExport" style="graphicPreferred"/>
```

37.20 opFilesSendTo

Die Operation FilesSendTo ruft das Menü zum Senden von verknüpften Dateien auf.

Beispiel

```
<Operation ID="SendTo" opCode="opFilesSendTo"/>
```

37.21 opFormEdit

Die Operation FormEdit ruft den Formular-Designer zum Beabeiten von Formularen auf. Sie unterstützt <Condition> **nicht**.

Beispiel

```
<Operation ID="FormEdit" opCode="opFormEdit" caption="Formular bearbeiten..."
          shortcut="Ctrl+Alt+F"/>
```

37.22 opFormEditText

Die Operation FormEditText zeigt das Formular als Text an.

Beispiel

```
<Operation ID="FormEditText" opCode="opFormEditText"
          caption="Formular als Text..." shortcut="Ctrl+Alt+T"/>
```

37.23 opImport

Die Operation Import ruft den Dialog zum Importieren von Objekte als Text, CSV, BIX oder SYM auf.

Die Option "Modifizierender Text-Import" steht nur für das Format Text oder CSV zur Verfügung.

Attribute:

supportSYM	Ermöglicht den Import von Condor-Symboldateien (*.sym), default = false
supportMOD	Blendet die Option "Modifizierender Text-Import" ein bzw. aus, default = true
supportBIX	Ermöglicht den Import von BIX-Dateien (*.bix), default = true

Funktioniert analog zur Operation Export.

Beispiel

```
<Operation ID="ImportText" opCode="opImport"
          supportSYM="true" supportMOD="true" supportBIX="true"/>
```

37.24 opLinkFiles

Die Operation LinkFiles wird zur Erzeugung von Verknüpfungen zu externen Datei benötigt.

Dafür werden Dateien aus dem Windows Explorer nach Byron/BIS z.B. in den Dokumenten-Explorer gezogen und losgelassen. Danach wird das Menü "Dokument hier erzeugen" aufgerufen.

Die Operation kann im Drag&Drop verwendet werden.

Beispiel

```
<Operation ID="LinkFiles" opCode="opLinkFiles" cursor="link"/>
```

Erweiterungen ab Byron/BIS v5.5.5:
Über das (optionale) Attribut `actionID` kann eine Aktion vom Typ *BIS-Funktion/Zuordnen von externen Dokumenten* zugeordnet werden.

Beispiel

```
<Operation ID="LinkFiles" opCode="opLinkFiles" actionID="LinkExtDoc" caption="Ext. Dokument verknüpfen" cursor="link"/>
```

Nach einem Drop werden die der Aktion zugewiesenen Schritte (ohne Dateiauswahl und Filter) ausgeführt (Bsp. Kopieren der Datei in anderes Verzeichnis o.ä.).

Ferner ist auch Drag&Drop im Format *CF_FileDescriptor* (OLE-Filestreams, E-Mail Anhänge aus **Microsoft Outlook** und anderen) möglich (nur wenn Aktion zugeordnet). Es ist zu beachten, dass in diesem Fall für ein korrektes Funktionieren eine Angabe von `TargetDir` in der Aktion zwingend ist!

Hinweis: Drag&Drop von E-Mail-Anhängen aus Microsoft Outlook ist nur mit **linker** Maustaste möglich (Einschränkung Microsoft Outlook nicht Byron/BIS)!

Konfigurationsbeispiel (Drag und Drop externer Dokumente):

Operation:

```
<Operation ID="AuftragDokumentZuordnenDragDrop" opCode="opLinkFiles"
           actionID="Instandhaltungsmanagement_LinkExtDoc"
           caption="Ext. Dokument verknüpfen">
  <Target>CLASS bisP_Order</Target>
</Operation>
```

DragDrop

```
<DragDrop keyState="-" external="true" cursor="copy">
  <Execute name=" AuftragDokumentZuordnenDragDrop "/>
</DragDrop>
```

37.25 opMove

Die Operation Move verschiebt die Source-Objekte unter das Target-Objekt des Application Elements.

Die Operation kann in Menüs, PopupMenüs und im Drag&Drop verwendet werden.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements oder das Drag-Objekt verwendet.

Als Eingangsobjekte (Input) für die `Target`-Filternavigation werden die Objekte in der Zwischenablage bzw. der Container oder die Drop-Objekte verwendet.

Beispiel

```
<Operation ID="MoveMenu" opCode="opMove">
    <Source>CLASS "bisB_MenuItem"</Source>
    <Target>[CLASS "bisB_MenuRoot" OR CLASS "bisB_MenuItem"]</Target>
</Operation>
```

37.26 opNavParameter

Diese Operation definiert einen Parameter für die Filternavigation. Dieser Parameter kann zwei Zustände annehmen: gesetzt (checked) oder nicht gesetzt (unchecked). Für den Zustand "checked" kann entweder ein Wert und/oder ein oder mehrere Objekte gesetzt werden. Für den Zustand "unchecked" kann nur ein Wert definiert werden. Der Parameter (paramName) kann in den Filternavigationen des Tools, z.B. bei anderen Operationen oder im Contents usw. verwendet werden. Um den Zustand des Parameters zu setzten, kann die Operation in einer ToolBar, einem Menüeintrag oder einem Popup-Menü aufgerufen (definiert) werden.

Die Operation unterstützt <Condition> **nicht**.

Enthaltene Attribute:

paramName	definiert den Parameter für die Filternavigation
uncheckedValue	definiert den Wert für den Zustand "unchecked"
checkedValue	definiert den Wert für den Zustand "checked"
checked	definiert den Standardzustand (true = "checked")

Enthaltene Elemente:

<Checked>	enthält die Filternavigation, welche die Objekte bei "checked" liefert hier kann auch das Attribut value angegeben werden, um zusätzlich einen Wert zu definieren

Beispiel 1 (liefert nur einen Wert im Zustand "checked")

```
<Operation ID="NavParameterA" opCode="opNavParameter"
           caption="Parameter A" paramName="nav_param_a"
           uncheckedValue="1" checkedValue="2">

</Operation>
```

Beispiel 2 (liefert einen Wert und Objekte bei im Zustand "checked")

```
<Operation ID="NavParameterB" opCode="opNavParameter"
           caption="Parameter B" paramName="nav_param_b"
           uncheckedValue="0">
    <Checked value="1">
        START CLASS Space
    </Checked>
</Operation>
```

37.27 opNavSubmenu

Diese Operation darf nur in einem Menu angewendet werden. Es wird dabei ein Untermenu aufgebaut, welches alle Objekte anzeigt, die mit <ToObjects> referenziert werden. Zwischen den Objekten der <ToObjects> wird jeweils eine Trennlinie eingefügt. Das entsprechende Objekt wird bei Menuclick geöffnet (‚Open')

Enthaltene Elemente:

<ToObjects> enthält die Filternavigation, welche zu den einzelnen Objekten führt.

<Display> enthält einen Ausdruck ($-Notation), welcher als Menu Text angewendet wird.
Default: `<Display>$Object_Display_Kontext$<Display>`

Beispiel

```
<Operation ID="RefDocuments" opCode="opNavSubmenu" caption="@mnDocuments_Caption@">
    <ToObjects>VIA Expl_Root_Of</ToObjects>
    <ToObjects>VIA Doc_Documents</ToObjects>
</Operation>
```

Vgl. opPropagateSubmenu

Anpassungen ab Byron/BIS v5.6.2: Über das Attribut `maxItemsDisplayed` kann gesteuert werden, wieviele Menüeinträge pro `<ToObjects>` maximal angezeigt werden (Vorgabe maximal 25 Einträge). Sind weitere Elemente vorhanden, werden diese unterdrückt und ein inaktiver Menüeintrag mit der Beschriftung "*Weitere Einträge vorhanden*" wird angezeigt.

37.28 opObjectRightsEdit

Die Operation ObjectRightsEdit ruf den Dialog zum Editieren der Objektrechte auf.

Beispiel

```
<Operation ID="ObjectRights" opCode="opObjectRightsEdit"/>
```

37.29 opOpen

Die Operation Open öffnet die Source-Objekte in einem Formular, falls vorhanden.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Ab v4.10.5: Das Attribut `forceNewWindow` bewirkt, dass Tool-Objekte (Objekte der Objektklasse `Doc_BisTool`) in einem neuen Fenster geöffnet werden auch wenn diese bereits geöffnet sind. So sind mehrere Fenster für dasselbe Tool möglich. `Default = false`.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="Open" opCode="opOpen"/>
```

```
<Operation ID="Open" opCode="opOpen" forceNewWindow="true">
    <Source>START "{99E2729A-A333-411D-9E30-1A66FEEF8F73}"</Source>
</Operation>
```

37.30 opOpenContainer

Die Operation OpenContainer öffnet das übergeordnete Objekt einers selektierten Objektes im Application Element, falls genau ein Objekt selektiert wurde.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="OpenContainer" opCode="opOpenContainer"/>
```

37.31 opPaste

Die Operation Paste kopiert bzw. verschiebt das Objekt aus der Zwischenablage (siehe opAddToClipboard). Dabei wird ein Drop auf das aktuelle (fokusierte) Application Element simuliert. Desshalb ist es notwendig, dass eine Drag&Drop-Operation für das Kopieren (opCopy) bzw. Verschieben (opMove) definiert wird (siehe Drag&Drop):

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="Paste" opCode="opPaste"/>
```

37.32 opPrint (erst ab Byron/BIS v4.10.3)

Die Operation Print ruft für das entsprechende Applikationselement die Seitenansicht bzw. das Drucken auf. Die Operation ersetzt das alte Print-Element, welches die Seitenansicht bzw. das Drucken von nur einem Applikationselement erlaubt. Bisher unterstützen nur die Elemente Grid, Gantt und Map (ab v5.0.3) die Operation.

Das Attribut `preview` gibt an, ob eine Seitenansicht (`true`) oder das Drucken (`false`) aufgerufen werden soll. Der Default ist `true`.

Attribute

`usePageDecoFrom`	Angabe, von welcher Printoperation die `PageDecoration` übernommen werden soll (ab v5.2.0).

Elemente

Das Element `PageDecoration` enthält die Einstellungen für die Seitenansicht im alten Byron/BIS-Format: (siehe auch Print-Element) Der Default entspricht dem Beispiel "Drucken mit Druckdefinition". Hinweis: Wenn ein die Druckdefinition angepasst werden soll, muss die komplette Druckdefinition angegeben werden!

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiele

Beispiel: Altes <Print> ersetzen (für Applikationselement Grid)

```
<Operations>
    <Operation ID="PrintPreview" opCode="opPrint" preview="true" forElement="grid"/>
    <Operation ID="Print" opCode="opPrint" preview="false" forElement="grid"/>
    …
</Operations>
```

```
<Menubar>
    <Menu name="MenuDocument">
        <Item name="mbPrintPreview" positionBefore="mbPreferences">
            <Execute name="PrintPreview"/> </Item>
        <Item name="mbPrint" positionBefore="mbPreferences">
            <Execute name="Print"/></Item>
    </Menu>
</Menubar>
```

Beispiel: Seitenansicht (für das aktuelle Applikationselement)

```
<Operation ID="PrintPreview" opCode="opPrint" preview="true">
    <PageDecoration>
        Orientation = landscape
        Font        = "Tahoma"
        … (vgl. Print-Element)
    </PageDecoration>
</Operation>
```

Beispiel: Drucken (für das aktuelle Applikationselement) – die PageDecoration wird von der Seitenansicht übernommen.

```
<Operation ID="Print" opCode="opPrint" preview="false" usePageDecoFrom="PrintPreview" />
```

Beispiel Drucken mit Druckdefinition (für das aktuelle Applikationselement)

```
<Operation ID="Print" opCode="opPrint" preview="false">
    <PageDecoration>
        Orientation = landscape
        Font        = "Tahoma"
        FontSize    = 10
        Margins
            left   = 20
            right  = 20
            top    = 20
            bottom = 15
        end
        Field:Database
            PagePos = left, top
            object  = user
            value   = "$bisB_UserToPerson%Object_Display_Kontext$"
            Y       = -1
        end
        Field:Database
            PagePos = center, top
            object  = container
            value   = "$Object_Display_Kontext$"
            Y       = -1
        end
        Field:text
            pagepos = left, bottom
            value   = "Datum: $Date$"
        end
        Field:text
            pagepos = right, bottom
            Value   = "Seite $PageNr$ von $TotalPages$"
        end
    </PageDecoration>
</Operation>
```

37.33 grPrinter

Die Operation grPrinter definiert den Ausdruck einer Grafik. Es gibt analog dazu den Befehl vgrPrint für VdrawGraphic

Wenn der Befehl aus dem Kontext eines <Graphic> Element verwendet wird, dann können diese EInstellung durch das Source Element weitergegeben werden:

<Source>[RECALL 'graphic.OBJ_StartObjects' OR RECALL 'graphic.OBJ_ViewObject']
</Source>

Die Konfiguration hat folgende Struktur:

```
<configuration>
  <batch>
  <massstab>
  <pane>
  <variable>
  <graphic>
  <decoration
  <layout>
</configuration>
```

Das Element `<batch>`

kommt höchstens einmal vor und definiert alle möglichen Startobjekte für den Stapeldruck. Das Eingansobjekt ist das Startobjekt und mit caption wird der Titel definiert, den die Checkbox erhält. Wenn der Nutzer die Checkbox anhakt, dann wird diese Filternavigation ausgeführt und für jedes Resultatobjekt ein Druck auf einer neuen Seite ausgeführt. Beispiel:

```
<batch caption="Alle Geschosse">
    VIA Object_To_Parent VIA Object_To_Children  CLASS Floor
</batch>
```

Das Element `<massstab>`

kommt höchstens einmal vor und definiert alle möglichen Massstäbe in ; getrennt.

Mite dem Attribut strict wird definiert, ob nur diese Werte zulässig sind (strict="true") oder ob diese Werte nur Vorschlagswerte sind, die vom Benutzer überschrieben werden können (strict="false")

Beispiel:

```
<massstab strict="true"> 100; 250; 500; 750; 1'000; 1'500; 2'000; 5'000 <massstab>
```

Das Element `<pane>`

kommt höchstens einmal vor und definiert alle möglichen (vordefinierte) Ausschnitte, zu welchen gesprungen werden kann.

Mit includeInitial="true" wird ein Eintrag für den Ausschnitt beim Aufruf erstellt.

Mit `includeInitial="true"` wird ein Eintrag für den Ausschnitt beim Aufruf erstellt.

Mit `includeExtent="true"` wird ein Eintrag für den Ausschnitt über alle Objekte (ohne Hintergrund) erstellt.

Weitere Ausschnitte können mit einem Element choice definiert werden. Dabei wird der Rahmen der Box in Meter (Modellkoordintaen) definiert.

<choice caption="Level" left="370" right="780" top="750" bottom="414"/>

Beispiel:

```
<pane includeInitial="true" includeExtent="true">
   <choice caption="Level" left="370" right="780" top="750" bottom="414"/>
</pane>
```

Das Element `<variable>`

enthält die die Definition aller Texte und Bitmaps, welche verwendet werden. Die Variablen werden dann in den Texten über eine $-Notation verwendet.

Beispiel:

Es wird eine Variable stw definiert, welche dann in einem Feld eingesetzt wird.

```
<variable>
  <txt name="stw">
    <nav attribute="bisB_SpaceNo">RECALL 'graphic.OBJ_StartObjects'
         VIA Object_To_Parent VIA Object_To_Parent
    </nav>
  </txt>
<variable>
```

Verwendung der Variable in einem Textfeld, in dem noch ein G vor den Wert gesetzt wird

```
<frame col="0" row="1" line="true" fill="true">
   <text caption="G$stw$" fontSize="0.6"/>
</frame>
```

Es gibt zwei Typen von Variablen: Bilder und Texte. In jedem Fall haben sie einen Namen, über den sie verwendet werden können.

Attribute von Element <txt>

Attribut	Beschreibung
name	Damit wird der Text in der $-Notation verwendet.
value	Damit wird der Typ festgelegt, es ist einer der Werte DAT, FRMT, INP, VAL_PARAM und MST. Die Bedeutung ist in der nächstten Tabelle erklärt.
title	Für value="INP": Das Eingabefeld wird so beschriftet.
comboWidth	Für value="INP": Das Eingabefeld bekommt diese Breite (in Piixel) default = „200"
store	Wenn dieses Attribut verwendet wird, dann werden die letzten 10 Werte des Benutzers in der Registry gespeichert. Der Wert von store ist der Schlüssel. Beispiel für store=titel": Computer\HKEY_CURRENT_USER\Software\Byron Informatik AG CH\bis\ Forms\TPrintGraphicForm_InpVals_Titel

Es gibt vordefinierte Variablenwerte, welche über value gewählt werden können. Wenn kein value angegeben wird, dann bestimmt eine Navigation <nav> mit einem Attribut den Wert.

Ein Bild wird entweder als Attribut an einem Objekt aus der Datenbank gelesen, oder als Referenz vom Dateisystem:

```
<pict name="pict" >
   <nav attribute="BSP_Bild" >
           RECALL 'graphic.OBJ_StartObjects' CLASS Space
           VIA Object_To_Parent VIA bisAR_SpaceToBuildingPart
   </nav>
</pict>
<pict name="pictX" value="VAL_PARAM" param="%temp%\bis\muster.jpg" />
```

Im ersten Fall ohne value, mit einer Navigation zum Objekt, welches das Bild im Attribut vom Typ Rastergrafik (in Binärdaten komprimiert) enthält.

Tipp:

Das Einlesen der Bilder in ein Attribut kann mit einer einfachen Filtternavigation erfolgen, indem der gesamte Dateipfad (Text) dem Rastergrafik Attribut zugewiesen wird:

```
START VALUE bisB_SpaceNo = 'B4' MODIFY BSP_Bild "D:\Data\Pict\b4.png"
```

Im zweiten Fall wird der Dateipfad durch Auswertung des `param` ermittelt.

Mit value können folgende Werte zugegriffen werden:

Wert von value	Beschreibung
DAT	Das aktuelle Datum. Die Formatierung erfolgt über die Variablen y m d h n s Beispiel: format="dd | mm | yyyy" Default: dd.mm.yyyy
FRMT	Papierformat: Die erste Ziffer bestimmt die Einheit der nachfolgenden Werte. 0=m, 1=dm, 2=cm, 3=mm Danach erfolgt das Format für Breite und Höhe Default: 2%.1f | %.1f Dies ergibt für A4 quer: 29.7 | 21.0
INP	Definiert ein Eingabefeld, in welchem der Benutzer Werte auswählen kann. Das Eingabefeld wird mit dem Wert von title beschriftet. Wenn am Schluss eine leere Wahl vorkommt, dann kann der Benutzer einen eigenen Text eingeben, sonst kann er nur einen Text der Auswahl selektieren. Die Breite der Eingabebox kann mit comboWidth verändert werden. Default: comboWidth="200"
MST	Der Massstab wird aus der Grösse der dargestellten Objekte (ohne Plan) ermittelt. Davon wird der nächst Grössere Wert aus `<massstab>` genommen, oder wenn dieser fehlt auf die nächsten 100 aufgerundet. Der Massstab ist eine ganze Zahl, im Format also ein %d. Default Format: format=" 1 : %d"
VAL_PARAM	Damit wird ein Parameter ausgewertet, den man in param angibt. Die Auswertung erfolgt über den OnParameter Mechanismus der FilterNavigation. Zum Beispiel die Ansicht der Graifk: param="graphic.VAL_CurrentView"

Beispiel für INP

```
<txt name="bem" comboWidth="300" value="INP" title="Bemerkung:">
  <choice caption="NUR ZUR INTERNEN VERWENDUNG"/>
  <choice caption="FREIGEGEBEN"/>
  <choice/> <!-- freie Eingabe auch möglich -->
```

```
</txt>
```

Das Element `<graphic>`

Mit diesem Element wird eine Grafik definiert, welche dann in einem Feld <frame> angezeigt werden kann. Wie bei den Variablen kann die Grafik über den name referenziert werden. Eine Grafik kann auch ohnne name sein und wird dann auch ohne name referenziert.

Die Attribute sind also `name` und `priorityAttribute`, beide sind optional.

Die enthaltenen Elemente sind start, contents und view. Diese beinhalten jeweils eine Filternavigation.

Beispiel:

```
<graphic>
  <start>RECALL 'graphic.OBJ_StartObjects' </start>
  <contents>BLOCK </contents>
  <view>RECALL 'graphic.OBJ_CurrentView' </view>
</graphic>
```

Das Start Objekt ist das Eingangsobjekt für contents und die view (Grafikansicht).

Mit der Grafikansicht (view) werden Objekte, Hintergrundsplan, Beschriftung und Färbung definiert. Wenn die Ansicht alle Objekte definiert (inputIs="startObject"), dann sollte der contents kein Objekte liefern.

Das Element `<decoration>`

Definiert das Aussehen eines Rechtecks mit einer bestimmten Grösse. Dieses kann dann in einem Layout an einer bestimmten Stelle eingesetzt werden. Inhaltlich sind decoration und layout identisch. Der Sinn der decoration liegt darin, dass sie von mehereren layouts verwendet werden kann. So kann man:

```
<decoration name="Kopf_A4Q_A3H" height="4.0" width="27.0">
```

In einem Layout für A4 quer und in einem Layout für A3 hoch verwenden. Die Attribute width und height werden jeweils in cm angegeben.

Das Element kann die Elemente defaults, cols, rows, und frame beinhalten.

Das Element `<layout>`

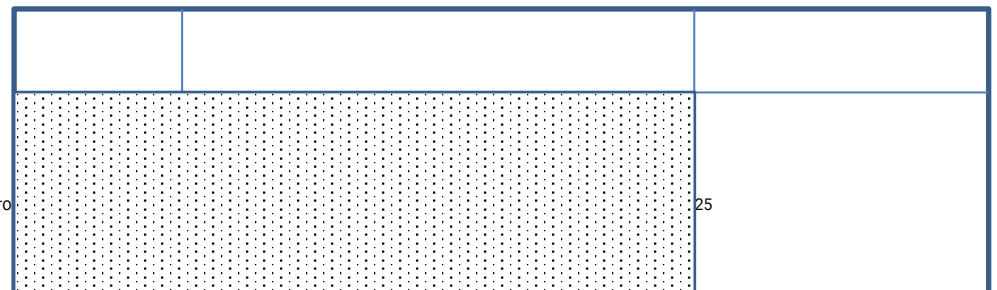
Ein layout definiert ein Aussehen für eine Papiergrösse. Der Benutzer wählt einen Drucker und kann dann einen zu diesem Papierformat passenden layout auswählen.

```
<layout caption="A4 simpel" format="A4" portrait="false">
```

Der Benutzer kann den Layout *`A4 simpel`* auswählen, wenn ein Drucker mit A4 ausgewählt wird. Das Blatt wird dann automatisch gedreht (`portrait="false"` bedeutet Querformat).

Flächenaufteilung `<cols> <rows> <frame>`

Die rechteckige Fläche, in der man sich befindet lässt sich in weiter rechteckige Fächen (frame) aufteilen. Es werden Spalten und Zeilen definiert



Angenommen wir befinden uns in Rechteck 15 cm Breit, 6 cm

```
<rows count="5"/>
<cols>
  <tab val="2.5"/>
  <tab val="10.0"/>
</cols>
<frame col="0" row="1" width="2" height="3" >
</frame>
```

Rows und cols werden durch Tabulatoren definiert. Die zwei Tabulatoren in cols definieren 3 Spalten. Der Wert in val wird in cm anggegeben. Bei rows wurde `count="5"` verwendet, damit werden 4 tab mit gleichen Abständen definiert.

Der frame wird durch die linke obere Ecke bei `col="0" row="1"` platziert. `width="2" height="3"` bezieht sich auch auf die Tabulatoren.

Defaults:

```
col="0" row="0" width="1" height="1"
```

Mit einem frame kann man Rahmen zeichen:

```
<frame col="0" row="0" width="3" height="5" line="true"/>
<frame col="2" row="0" width="0" height="5" line="true"/>
```

Im zweiten Fall wird kein Rechteck, sondern nur ein senkrechter Strich beim zweiten Tabulator gezeichne

Ein frame kann eingefärbt werden:

```
<frame line="true" fill="true" />
```

Inhalt von frame `<picture> <text> <graphic> <legend>`

Ein frame kann einen Text beinhalten, hier wird noch die Variable `geb` eingesetzt.

```
<frame>
  <text caption="GEBÄUDE $geb$"/>
</frame>
```

Ein Text beinhalten eine Expression, wenn der Text mit einem = beginnt. Im Folgenden gibt es eine Variable Ansicht, diese wird in Grossbuchstaben umgewandelt.

```
<frame>
```

```
  <text caption="= STRCASE(ansicht, 1)"/>
</frame>
```

Attribute von Element `text`:

Attribut	Beschreibung
`hAlign`	Die horizontale Ausrichtung hat einen der folgenden Werte: `left, center, right` Default: `top`
`vAlign`	Die vertikale Ausrichtung hat einen der folgenden Werte: `top, center, bottom` Default: `left`
`textMultiline`	Merhzeiliger Text: `false, true` Default `false`
`textColor`	RGB Wert in der Form #D0D0D0 (das ist ein hellgrau) Default: #000000 (schwarz)
`fontSize`	Gösse (Höhe) der Schrift in cm Default: 0.35
`fontWeight`	Das Gewicht kann mit Werten von 1.0 bis 9.0 definiert werden: 1.0 thin, 2.0 extra light, 3.0 light, 4.0 normal, 5.0 medium, 6.0 semi bold, 7.0 bold, 8.0 extra bold, 9.0 heavy Default: 4.0
`fontName`	Fontname Default: Arial

Ein frame kann eine Rastergrafik beinhalten hier das `<pict name="pict" >`

```
<frame>
  <picture pictName="pict"/>
</frame>
```

Ein frame kann eine (die) Grafik beinhalten, hier diejenige ohne `name`

```
<frame>
  <graphic/>
</frame>
```

Ein frame kann eine Legende beinhalten

```
<frame>
  <legend width="3.5" height="0.3" margin="0.04"/>
</frame>
```

Zusätzliche Attribute der Legende

Attribut	Beschreibung
`height`	Zeilenabstand, wenn dieser mit 0 definiert ist, dann wird die doppelte Font-grösse genommen. Default: `0 wird zu 2 Mal Fontgrösse`

`margin`	Definiert den Rand innerhalb einer Zeile. Das gefärbte Quadrat hat damit die Seitenlänge von: height – (2 * margin). Default: `0.1`
`width`	Spaltenbreite. Mit 0 wird die Spaltenbreite auf Breite des übergeordneten Frame gesetzt, d.h. die Leggende wird einspaltig. Wenn der Frame eine Breite von 10.0 hat und width 3.0, dann ist die Legende 3-spaltig Default `0 (einspaltig)`

Atttribute und Vorgaben

Die folgenden Attribute können an verschiedenen Elementen vorkommen, typischerweise an frame und text. Der Wert eines Attributes wird ‚vererbt', muss also in einem unterliegenden Element bei Bedarf wieder zurück gesetzt werden. Mit defaults können die Werte allgemein gesetzt werden.

Beispiel:

```
<defaults>
   <default fontName="Verdana"/>
   <default fontSize="0.25"/>
   <default lineColor="#000000"/>
   <default interiorColor="#D0D0D0"/>
   <default vAlign="center"/>
 </defaults>
```

Attribute:

Attribut	Beschreibung

`lineColor`
`interiorColor`
`textColor`
	Farben werden als RGB Wert in der Form #000000 (das ist schwarz)

`lineWidth`	Liniendicke in cm, wobei 0 eine Standard Linie ist. Default: 0

`marginLeft`
`marginRight`
`marginBottom`
`marginTop`
	Rand in cm
	Default: 0

`hAlign`	Die horizontale Ausrichtung hat einen der folgenden Werte: `left, center, right` Default: `top`
`vAlign`	Die vertikale Ausrichtung hat einen der folgenden Werte: `top, center, bottom` Default: `left`
`textMultiline`	Merhzeiliger Text: `false, true` Default `false`
`textColor`	RGB Wert in der Form #D0D0D0 (das ist ein hellgrau) Default: #000000 (schwarz)
`fontSize`	Gösse (Höhe) der Schrift in cm
`fontWeight`	Das Gewicht kann mit Werten von 1.0 bis 9.0 definiert werden: 1.0 thin, 2.0 extra light, 3.0 light, 4.0 normal, 5.0 medium, 6.0 semi bold, 7.0 bold, 8.0 extra bold, 9.0 heavy

	Default: 4.0
`fontName`	Fontname Default: Arial

37.34 vgrPrint

Dies ist die analoge Operation zu grPrinter, aber für VdrawGraphic Elemente.

Der Aufbau sieht so aus:

```
<Operation ID="GPrint" opCode="vgrPrint" icon="opPrint" forVdrawGraphic="graphic">
     <Definition>
          <variable>
          </variable>
          <textSetting />
          <layout format="A4" landscape="true">
          </layout>
          <layout format="A4" landscape="flase">
          </layout>
          <layout format="A3" landscape="true">
          </layout>
          <layout format="A3" landscape="false">
          </layout>
     </Definition>
</Operation>
```

Das Element `<Definition>`

Enthält die Elemente variale textSetting und für jedes Papierformat ein layout.

Mit den Attributen horMargin und verMargin kann der Rand (in cm) definiert werden.

Default: `<Definition horMargin= "0.3" verMargin= "0.3">`

Das Element `<variable>`

enthält die die Definition aller Texte und Bitmaps, welche verwendet werden. Die Variablen werden dann in den Texten über eine $-Notation verwendet.

Beispiel:

Es wird eine Variable stw definiert, welche dann in einem Feld eingesetzt wird.

```
<variable>
  <txt name="stw">
    <nav attribute="bisB_SpaceNo">RECALL 'graphic.OBJ_StartObjects'
         VIA Object_To_Parent VIA Object_To_Parent
    </nav>
  </txt>
<variable>
```

Verwendung der Variable in einem Textfeld, in dem noch ein G vor den Wert gesetzt wird

```
<frame col="0" row="1" >
```

```
   <text caption="G$stw$"/>
</frame>
```

Dazu gibt es die vordefinierten Variablen format, now und scale. Optional kann nach diesen Variablen ein Format definiert werden, durch einen : getrennt.

Es gelten folgende Defaults:

$format$	$format :A%0:d$
now	$now:dd.mm.yyyy$
$scale$	$scale:%.0f$

Bei format gibt es noch zwei weitere Argumente: Breite (%1:) und Höhe (%2:) in Millimeter.
Beispiel:

$format: %1:.0f \| %2:.0f$	(für DIN A4 hoch: "210 | 297")
$format :A%0:d$	(für DIN A4 hoch: "A4")

Parameter aus dem Kontext der Filteravigation könne auch direkt mit der $-Notation zugegriffen werden, z.B die Ansicht aus der Grafik: $graphic.VAL_CurrentView$

Es gibt zwei Typen von Variablen:
Texte <txt> und Bemerkungen <rem>. In jedem Fall haben sie einen Namen, über den sie verwendet werden können.

```
Beispiel:
          <variable>
            <txt name="vrsn" format="ddmmyy">
              <nav attribute="CIR_PlanungsversionDatum">
                VIA CIR_Planversion2Planungsversion
              </nav>
            </txt>
            <rem name="plantitel" title="Plantitel:" store="Titel">
                <choice caption="GESCHOSSPLAN VERMARKTUNG"/>
                <choice/> <!-- freie Eingabe auch möglich -->
            </rem>
            <rem name="bem" comboWidth="300" store="Bem" title="Bemerkung:">
               <choice caption="NUR ZUR INTERNEN VERWENDUNG"/>
               <choice/> <!-- freie Eingabe auch möglich -->
            </txt>
            <txt name="leg">
              <nav attribute="name">RECALL 'coloring'</nav>
            </txt>
            <txt name="titA">
               <nav attribute="CIR_Plantitel1">
                   RECALL 'graphic.OBJ_StartObjects'
                   [CLASS CIR_Planversion OR CLASS CIR_PlanversionLevel]
               </nav>
            </txt>
            <txt name="titAkurz">
               <nav attribute="CIR_Plankurztitel">
                   RECALL 'graphic.OBJ_StartObjects'
                   [CLASS CIR_Planversion OR CLASS CIR_PlanversionLevel]
```

```
      </nav>
    </txt>
    <txt name="stw">
      <nav attribute="bisB_SpaceNo">
        RECALL 'graphic.OBJ_StartObjects'
        [CLASS CIR_PlanversionLevel VIA Object_To_Parent
        OR CLASS CIR_Planversion VIA Object_To_Parent
        VIA Object_To_Parent]
      </nav>
    </txt>
    <txt name="geb">
      <nav attribute="Name">RECALL 'graphic.OBJ_StartObjects'
           VIA Object_To_Parent
      </nav>
    </txt>
  </variable>
```

Das Element `<textSetting>`

enthält die Definition für die Erscheinung der Texte:

```
<textSetting horMargin="0.5" verMargin="0.1"
             font='Verdana' size="0.5"
             verAdjust="Bottom" horAdjust="Left" />
```

Das Element definiert die Erscheinung der Texte. Das Element kann in <Defintiion>, <layout> oder <frame> vorkommen. Es wird jeweils die Definition des übergeordneten überschrieben, bzw. den spezifizierten Teil geändert.

Attribute von <textSetting> und <text>

Attribut	Beschreibung
`horMargin` `verMargin`	Horizontaler bzw. vertikaler Rand in cm
`size`	Textgrösse in cm
`bold` `italic`	Fett bzw. kursive. Mögliche Werte zB, false, true oder 0, 1
`widthFactor`	Float. Default 1.0
`font`	Text-Font zB. `'Verdana' vgl. C:\Windows\Fonts`
`verAdjust`	`'BaseLine', 'Bottom', 'Center', 'Top'`
`horAdjust`	`'Left', 'Center', 'Right'`

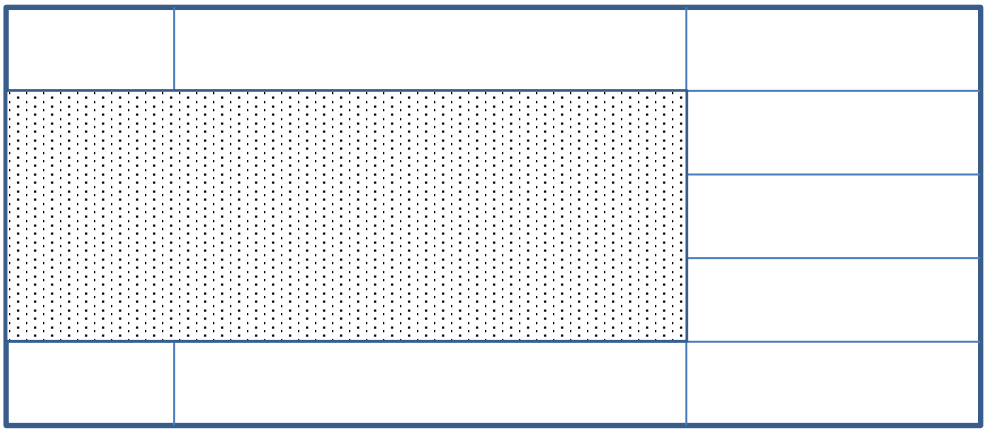
Das Element `<layout>`

enthält die die Definition einer Seite. Für jedes Format wird ein Layout definiert.

Ein layout besteht aud einem oder mehreren <frame>, welche wiederum <frame> enthalten können. Der layout ist der äusserste frame, frames und layouts können daher dieselben Elebente enthalten.

<rows> und <cols> definieren das Grundraster in einem frame, bzw, im layout:

Es werden Spalten und Zeilen definiert



Angenommen wir befinden uns in einem Rechteck 15 cm breit und 6 cm hoch

```
<rows count="5"/>
<cols>
  <tab val="2.5"/>
  <tab val="10.0"/>
</cols>
<frame col="0" row="1" width="2" height="3" >
</frame>
```

Der frame umfasst die punktierte Fläche.

Zu den Tabpositionen kann noch ein Rand (in cm) definiert werden:

```
marginLeft, marginRight, marginBottom marginTop
```

Beispiel:

```
<frame col="1" row="0" line="true" >
  <rows count = "8"/>
  <cols>
    <tab val="2.0"/>
  </cols>
  <textSetting size="0.25" />
  <text   col="0" row="0" caption="Gebäude:"/>
  <text   col="1" row="0" caption="$geb$"/>
  <text   col="0" row="1" caption="Stockwerk:"/>
  <text   col="1" row="1" caption="$stw$"/>
  <text   col="0" row="2" caption="Massstab:"/>
  <text   col="1" row="2" caption="1 : $scale:$"/>
  <text   col="0" row="3" caption="Datum:"/>
  <text   col="1" row="3" caption="$now:dd | mm | yyyy$"/>
  <text   col="0" row="4" caption="Format:"/>
  <text   col="1" row="4" caption="$format: %1:.0f | %2:.0f$"/>
  <text   col="0" row="5" width="2" caption="$bem$" size="0.3" horAdjust="Center"/>
  <picture col="0" row="6" width="2" height="3" path="%appdata%\byron\logo.jpg"/>

  <frame col="1" row="0" width="0" height="4" line="true" />
```

```
  <frame col="0" row="1" width="2" height="0" line="true" />
  <frame col="0" row="2" width="2" height="0" line="true" />
  <frame col="0" row="3" width="2" height="0" line="true" />
  <frame col="0" row="4" width="2" height="0" line="true" />
  <frame col="0" row="5" width="2" height="0" line="true" />
</frame>
```

Das Element `<frame>`

Attribut	Beschreibung
`fill`	Füllfarbe als RGB, Bsp: `"#e0e0e0"`
`line`	Rand mit Linie. Mögliche Werte zB, false, true oder 0, 1

Das Element <graphic>

Hier wird die Grafik gezeichnet. Es gibt keine spezifischen Attribute

Das Element <legend>

Hier wird die Legende der Grafik platziert. Dieses Element kann auch alle Attribute von `<textSetting>` haben

Attribut	Beschreibung
`line`	Default 1.8 Zeilenhöhe, gemessen an der Textgösse (1.0 = Texthöhe)
`cols`	Default 1 Anzahl Spalten
`hdist`	Abstand zwischen Farbrechteck und Text
`wdist`	Breite des Rechtecks (1 = gleich wie Höhe)

Das Element <text>

Dieses Element kann auch alle Attribute von `<textSetting>` haben

Attribut	Beschreibung
`caption`	Text der ausgegeben wird, dieser kann in $ Notation auch Werte von Variablen übernehmen.

Das Element <picture>

Dieses Element enthält ein Bild (Bitmap) welches über 'path' referenziert wird, oder sich in der Datenbank in einem Attribut befindet. Das Bild wird eingemittet, unter Einhaltung des Verhältnis Höhe zu Breite.

Attribut	Beschreibung
`horMargin`	Default: 0.02 linker und rechter Rand in cm nach Innen.
`verMargin`	Default: 0.02 oberer und unterer Rand in cm nach Innen.
`path`	Pfad auf die Bilddatei. Wenn dieser angegeben ust, wird ein enthaltenes <nav> Element ignoriert.

Enthaltenes Element <nav> Beispiel mit einem binären Attribut `"BSP_Bild"`

```
<picture>
  <nav attribute="BSP_Bild" >
    RECALL 'graphic.OBJ_StartObjects'
  </nav>
</picture>
```

Tipp:

Das Einlesen der Bilder in ein Attribut kann mit einer einfachen Filtternavigation erfolgen, indem der gesamte Dateipfad (Text) dem Rastergrafik Attribut zugewiesen wird:

```
START VALUE bisB_SpaceNo = 'B4' MODIFY BSP_Bild "D:\Data\Pict\b4.png"
```

37.35 evdPrint

Dies ist die analoge Operation zu grPrinter und vgrPrint, aber für ExternalVectordraw-Elemente.

Der Befehl enthält ein Element **Definition** und beliebig viele Elemente **Variable**. Die Definition kann Variablen enthalten, diese werden so lange ersetzt, bis keine Variable mehr gefunden wird.

37.35.1 Element `<Variable>`

Eine Variable enthält eine Filternavigation, welche zum Objekt führt, das verwendet wird. Eine Ausnahme ist das Attribut NOW, das wird ohne Objekt ausgewertet.

Attribute vom Element Variable:

Attribut	Beschreibung
name	Dieser Name wird dort verwendet, wo der Wert eingesetzt werden soll.
attribute	ByronBIS Attribut, welches als Wert verwendet wird. speziell: "NOW", bezeichnet das aktelle Datum und Uhrzeit.
format	Damit kann eine Formatierung angegeben werden, die von der Vorgabe aus ByronBIS abweicht.

Beispiele:

```
<Variable name="({-Ebene-})" attribute="bisB_SpaceId">
  RECALL 'XvectorDraw.OBJ_StartObjects'
</Variable>
<Variable name="({-Flaeche-})" attribute="bisB_NetGroundArea" format="%.0f m²">
  RECALL 'XvectorDraw.OBJ_StartObjects'
</Variable>
<Variable name="({-FlOhneFormat-})" attribute="bisB_NetGroundArea" >
  RECALL 'XvectorDraw.OBJ_StartObjects'
</Variable>
<Variable name="({-Jetzt-})" attribute="NOW" format="d. mmm yyyy"/>
<Variable name="({-Zeit-})" attribute="NOW" format="hh:nn"/>
<Variable name="({-Ansicht-})" attribute="Name">
  RECALL 'XvectorDraw.OBJ_CurrentView'
</Variable>
```

Da die Variablen sol lange ausgewertet werden, bis keine mehr gefunden wird, können die Definition auch in die Datenbank (z.B. als `Doc_Note`) ausgelagert werden:

```
<Variable name="({-A4H-})" attribute="Doc_Note_property">
                START VALUE Doc_Id = Print_A4H
</Variable>
<Variable name="({-A4Q-})"attribute="Doc_Note_property">
                START VALUE Doc_Id = Print_A4Q
</Variable>
<Variable name="({-Ax-})"attribute="Doc_Note_property">
                START VALUE Doc_Id = Print_Ax
</Variable>
<Definition>({-Ax-})</Definition>
```

37.35.2 Element `<Definition>`

<Definition> JSON-Script, welches den Aufbau der Seite definiert. Es wird unter Konfiguration => Alle Dokumente => Dokumente => Abgleiche => Flächenmanagement => PrintDefinition konfiguriert

Attribut	Beschreibung
`A4H`	A4-Hochformat
`A4Q`	A4-Querformat
`NumOfXTabs` `NumOfYTabs`	Definiert ein Raster über die Seite mit der abgegebenen Anzahl an Spalten und Zeilen.
`PageBorderSize`	Die Grösse des Seitenrands in mm. Falls der Rand des Druckertreibers grösser ist als der hier definierte, wird der des Druckertreibers verwendet. Ab Verion 5.5.9
`StartX` `StartY`	Wird von jedem «-Rect»-Element und allen TextObjects benötigt. Erstes Raster auf der X-Achse/Y-Achse, das verwendet werden soll.
`Width`	Wird von jedem «-Rect»-Element benötigt. Anzahl Raster auf der X-Achse, welche verwendet werden sollen
`Height`	Wird von jedem «-Rect»-Element benötigt. Anzahl Raster auf der Y-Achse, welche verwendet werden sollen
`HideBorder`	«`true`» zum Ausblenden des Randes. Zulässig in allen «-Rect»-Elementen und den TextObjects. Ab Version 5.5.9
`Text`	Statischer und/oder dynamischer Text.
`VectorDrawRect`	Rechteck, in welchem der Plan angezeigt wird.
`TextObjects`	Erwartet eine Liste von Rechtecken, in denen Textelemente angezeigt werden. Jedes Element braucht mindestens die Attribute «StartX/Y», «Width», «Height» und «Text».
`TextSize`	Die Schriftgrösse in % von der Höhe des zugehörigen Objekts. (Ab Version 5.5.8) Die Schriftgrösse in Pt. (Ab Version 5.5.9) Zulässig in TextObjects, MassstabRect und CommentRect.
`IsBold`	Die Schrift in fett, falls auf «true» gesetzt. (Ab Version 5.5.10)

	Zulässig in TextObjects, MassstabRect und CommentRect.
`IsItalic`	Die Schrift in kursiv, falls auf «true» gesetzt. (Ab Version 5.5.10) Zulässig in TextObjects, MassstabRect und CommentRect.
`Font`	Die Schriftart, als Pfad angegeben. Bspw. "C:\\Windows\\Fonts\\Comic Sans MS". (Ab Version 5.5.10) Zulässig in TextObjects, MassstabRect und CommentRect.
`Color`	Die Farbe des Texts angegeben als RGB Code als Integer. (Ab Version 5.5.10) Zulässig in TextObjects, MassstabRect und CommentRect.
`ImageObjects`	Erwartet eine Liste von Rechtecken mit je einem «ImagePath» und zeichnet das Bild im angegebenen Rechteck. Das Bild wird so passend gemacht, dass es ins Rechteck passt.
`ImagePath`	Der Pfad zum Ort, an welchem das Bild liegt.
`LegendRect`	Rechteck, in dem die Legende angezeigt wird. Ab Version 5.5.9
`ComponentLegendRect`	Rechteck, in dem die Komponentenlegende angezeigt wird. Ab Version 5.5.10
`MaxTextLength`	Nur im LegendRect zulässig. Definiert die maximale Textlänge der Legendetexte. Alles was länger ist wird abgeschnitten und durch «…» ersetzt.
`MassstabRect`	Rechteck, in dem der Massstab angezeigt wird. Ab Version 5.5.9
`CommentRect`	Rechteck, in das ein Kommentar geschrieben werden kann. Ab Version 5.5.9

Beispiel


```
"_comments": "({-Anwender-}),({-Gebaeude-}), ({-Ebene-}), ({-Flaechenausmass-}),
({-JetztDatum-}), ({-JetztZeit-}), ({-VerwendeteAnsicht-}), ({-LogoPath-})",
"A4Q": {
   "NumOfXTabs": 60,
   "NumOfYTabs": 40,
   "PageBorderSize": 10,
   "VectorDrawRect": {
      "StartX": 0,
      "StartY": 2,
      "Width": 60,
      "Height": 35
   },
   "LegendRect" : {
      "StartX" : 0,
      "StartY" : 2,
      "Width" : 9,
      "Height" : 7,
      "HideBorder" : true,
      "MaxTextLength" : 8
   },
   "MassstabRect" : {
      "StartX" : 23,
      "StartY" : 0,
      "Width" : 17,
      "Height" : 2,
      "TextSize" : 12,
      "IsItalic": true
   },
   "CommentRect" : {
      "StartX" : 50,
      "StartY" : 2,
      "Width" : 10,
      "Height" : 15,
      "TextSize" : 8,
      "IsItalic": true,
      "HideBorder" : true
   },
   "ImageObjects": [
      {
         "StartX": 0,
         "StartY": 39,
         "Width": 60,
         "Height": 1,
         "ImagePath": "({-LogoPath-})",
         "HideBorder" : true
      }
   ],
   "TextObjects": [
      {
```

```
    "StartX": 0,
    "StartY": 37,
    "Width": 16,
    "Height": 2,
    "TextSize": 12,
    "HideBorder" : false,
    "IsBold": true,
    "IsItalic": true,
    "Font": "C:\\Windows\\Fonts\\Comic Sans MS",
    "Color": 340735,
    "Text": "Geb: ({-Gebaeude-})"
},
{
    "StartX": 16,
    "StartY": 37,
    "Width": 24,
    "Height": 2,
    "TextSize": 12,
    "Text": "Stw: ({-Ebene-})"
},
{
    "StartX": 40,
    "StartY": 37,
    "Width": 20,
    "Height": 2,
    "TextSize": 12,
    "Text": "Ansicht: ({-VerwendeteAnsicht-})"
}
,
{
    "StartX": 0,
    "StartY": 0,
    "Width": 16,
    "Height": 2,
    "TextSize": 12,
    "Text": "({-Anwender-})"
}
,
{
    "StartX": 16,
    "StartY": 0,
    "Width": 7,
    "Height": 2,
    "TextSize": 12,
    "Text": "({-Flaechenausmass-})"
}
,
{
    "StartX": 40,
    "StartY": 0,
    "Width": 20,
    "Height": 2,
    "TextSize": 12,
```

```
            "Text": "({-JetztDatum-}); ({-JetztZeit-})"
        }
    ]
}
```

37.36 opPropagate

Diese Operation führt ein Propagate zu einem Applications Element aus (siehe Propagate).

Attribute:

toApplication	ID der Application für das Propagate (optional, erst ab v4.8.0). Default ist die aktuelle Application, falls keine angegeben wird.
toElement	ID des Application Elements, welches das Propagate erhält.
propagateName	Propagate Name (optional), wird nur bei propagateType = "`custom`" übergeben (entspricht dem Attribut nameTarget, siehe Propagate) (default = fromParameter)
fromParameter	Parameter-Wert für das Propagate. Kann entweder einen Parameterwert (z.B. VAL_CurrentDate) einen konstanten Wert ('Hallo', '17', '8.1 47.3') oder ein Objekt (z.B. Grid.OBJ_Selection) enthalten. Wenn ein Wert verwendet wird und der propagateType = "`custom`" ist, wird fromParameter in `additional.Data` an den Empfänger übergeben. Ab V 5.0.0 optional
propagateCode	Propagate Code (optional) , wird nur bei propagateType = "`custom`" übergeben (siehe Propagate)
propagateType	Propagate Typ: "`load`" ¦ "`select`" ¦ "`show`" ¦ "`custom`" (default = "`custom`") **N.B.**: "`show`" wird erst ab Version v4.11.3 unterstützt.

Enthaltene Elemente:

<Tool>	Filternavigation zur Ermittlung des Tool bzw. Explorers (optional). Default ist das aktuelle Tool, falls keines angegeben wird. Es werden keine Eingangsobjekte (Input) übergeben.
<Source>	Filternavigation für Eingangsobjekte (optional).
<Condition>	Filternavigation für eine Bedingung (optional).
<Transform>	Filternavigation, welche auf das Objekt vor dem Propagate angewendet wird (optional) Als Eingangsobjekte (Input) werden die gefundenen-Objekte aus dem "fromParameter" verwendet.

Beispiel "Doppelklick auf ein selektiertes Objekt im Grid selektiert auch das Objekt im Baum":

```
<Operation ID="SelectParentInTree"
           opCode="opPropagate"
           forElement="grid"
           toElement="tree"
           propagateType="select">
    <Condition>VIA Room_To_Component</Condition>
    <Transform>VIA Room_To_Component</Transform>
</Operation>
…
<Grid ID="grid" panel="pnQualityGrid" onDoubleClick="SelectParentInTree">
    …
</Grid>
```

Vor `Version 5.0.0` funktionierte `forElement="grid"` nicht und musste durch `fromParameter="grid.OBJ_Selection"` ersetzt werden.

Beispiel "Sendet das Datum an ein Formular":

```
<Operation ID="SendDateToForm" opCode="opPropagate"
           toApplication="ApplExplorer"
           toElement="form"
           propagateType="custom"
           propagateName="Date"
           propagateCode="12345"
           fromParameter="planner.VAL_CurrentDate"/>
```

Beispiel "Sendet den Zoomfaktor 14 an eine Map":

```
<Operation ID="SendZoom14ToMap" opCode="opPropagate"
           toApplication="ApplExplorer"
           toElement="Map"
           propagateType="custom"
           propagateName="zoom"
           fromParameter="'14'"/>
```

37.37 opPropagateSubmenu

Diese Operation darf nur in einem Menu angewendet werden. Es wird dabei ein Untermenu aufgebaut, welches alle Objekte anzeigt, die mit <ToObjects> referenziert werden. Zwischen den Objekten der <ToObjects> wird jeweils eine Trennlinie eingefügt. Das entsprechende Objekt wird bei Menuclick das Objekt ‚propagiert' für die Einstellungen des propagierens vgl. opPropagate. Die Operation ist eine Kombination von opPropagate und opNavSubmenu

Enthaltene Elemente:

<ToObjects>	enthält die Filternavigation, welche zu den einzelnen Objekten führt.
<Condition>	Bedingung (vgl. opPropagate).
<Transform>	wird auf das Objekt vor dem propagate angewendet (vgl. opPropagate).
<Display>	enthält einen Ausdruck ($-Notation), welcher als Menu Text angewendet wird.
Default:	`<Display>$Object_Display_Kontext$<Display>`

Attribute:

Caption Menubezeichnung

Inline default="false" Mit true wird kein Untermenu, sondern die Menüeinträge erfolgen gleich im Menu.

Beispiel

```
<Operation ID="propagateSubMenuTest" opCode="opPropagateSubmenu"
           caption="propagate Beispiel"
           toElement="grid"
           propagateType="load">
    <!--
    <Condition>CLASS Space</Condition>
    <Transform>VIA Object_To_Parent</Transform>
    -->
    <Display>zu $Object_Display_Kontext$</Display>
    <ToObjects>VIA Object_To_Children</ToObjects>
    <ToObjects>VIA Object_To_Parent</ToObjects>
</Operation>
```

Vgl. opNavSubmenu und opPropagate

37.38 opProperties

Die Operation Properties öffnet die Source-Objekte im Eigenschaftsblatt.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="Properties" opCode="opProperties"/>
```

37.39 opResetAttribute

Die Operation ResetAttribute setzt die Vorgabewerte für das aktuelle Attribut des Application Elements für alle selektierten Objekte.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden und für Applikationselemente, welche ‚Attributselektion' unterstützen, zB. Grid.

Beispiel

```
<Operation ID=" ResetAttribute " opCode="opResetAttribute"/>
```

Mit name kann ein Atrribut für die Operation definiert werden, damit funktioniert die Operation für alle Appplikationselemente.

Beispiel mit Name

```
<Operation ID=" ResetAttribute" opCode="opResetAttribute" name="bisB_AreaGeometrie"/>
```

37.40 opSearch

Die Operation Search ruft die Objekt-Suche auf, mit welcher kontextsensitiv oder global nach Objekten gesucht werden kann. Dabei wird die Selektion (falls vorhanden) oder der Container des Application Elements als Startobjekt für die Suche verwendet.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel kontextsensitiv

```
<Operation ID="Search" opCode="opSearch" forElement="tree"/>
```

Beispiel global

```
<Operation ID="Search" opCode="opSearch" searchDatabase="true"/>
```

37.41 opSelectAll

Die Operation SelectAll selektiert alle Objekte eines Application Eelements.

Die Operation kann nur in Menüs und PopupMenüs verwendet werden.

Beispiel

```
<Operation ID="SelectAll" opCode="opSelectAll" onElement="grid" shortcut="Ctrl+A"/>
```

37.42 opShellExecute (erst ab Byron/BIS v4.7.7)

Die Operation ShellExcute führt einen Befehl der Windows Shell aus.

Attribute:

params	Eine Formel in $-Notation, welche für alle Objekte ausgewertet und zusammengesetzt wird. Der resultierende Text wird als Parameter mit dem Command übergeben.
command	Der Shell-Befehl mit allfälligen Parametern. Der Text "$$" wird durch die ausgewertete params-Formel ersetzt.

Beispiel

```
<Operation ID="SendMail" opCode="opShellExecute" caption="E-Mail senden..."
          command="mailto:$$" params="$bisB_Email$;"/>
```

37.43 opSequence (erst ab Byron/BIS v5.0.1)

Die Operation `opSequence` führt eine Reihe anderer Operationen aus.

Das `allConditions`-Attribut gibt an, ob zur Ausführung der Operation eine erfolgreiche Prüfung *aller* zugehörigen Operations nötig ist (default `false`).

Bei Ausführung der Operation werden die einzelnen angegebenen Operationen aus den Executes nacheinander, in eigenen Transaktionen ausgeführt.

Beispiel

```
<Operation ID="MySequence" opCode="opSequence" allConditions="true">
    <Condition>
        CLASS Room
    <Condition>
    <Execute name="Op1"></Execute>
    <Execute name="Op2"></Execute>
    <Execute name="Op3"></Execute>
</Operation>
```

Die Operation `opSequence` kann in **Drag & Drop** ebenfalls verwendet werden (ab Byron/BIS **v5.3.2**). Dabei kann in den aufgerufenen Operationen auf Source und Target der Drag&Drop-Operation zugegriffen werden, indem OBJ_Source und OBJ_Target, der Sequence verwendet wird:
`<Sequence-ID>".OBJ_Source"` bzw. `<Sequence-ID>".OBJ_Target"`

Beispiel: Ablegen von Objekten in einem Grid mit Drag & Drop

```
<!-- aus der Sequence aufgerufene Operation -->
<Operation ID="LoadGridWithDD" opCode="opPropagate" toElement="grid" propagateType="load">
    <Transform>RECALL "LoadGridWithDDSeq.OBJ_Source"</Transform>
</Operation>

<Operation ID="LoadGridWithDDSeq" opCode="opSequence">
    <Source>
        -- vgl. opExecuteFilterNavigation
    </Source>
    <Target>
        -- vgl. opExecuteFilterNavigation
    </Target>
    <Condition>
        -- vgl. opExecuteFilterNavigation
    <Condition>
    <Execute name="LoadGridWithDD"></Execute>
</Operation>
```

37.44 opShow

Die Operation Show zeigt (sucht und selektiert) die Source-Objekte in dem entsprechendem Application Element.

Die Operation kann in Menüs, PopupMenüs und im Drag&Drop verwendet werden.

Als Eingangsobjekte (Input) für die `Source`-Filternavigation werden die Objekte in der Zwischenablage oder die Drag-Objekte verwendet.

Als Eingangsobjekte (Input) für die `Target`-Filternavigation wird die Selektion des Application Elements oder das Drop-Objekt verwendet.

Mit `onSourcePropagate` bzw. `onTargetPropagate` wird gesteuert, ob die zu zeigenden Objekte bei den über Propagates verbundenen Applikationselementen (des Target-Applikationselements) gesucht werden sollen. `onSourcePropagate` steuert dies für die eingehenden Propagates, `onTargetPropagate` für die ausgehenden Propagates. Es wird empfohlen, beide Werte immer auf true zu setzen.

Beispiel

```
<Operation ID="Show" opCode="opShow"
           onSourcePropagate="true"
           onTargetPropagate="true"/>
```

37.45 opShowHistory

Die Operation ShowHistory zeigt die Änderungshistorie an.

Attribute:

onlyIfTraced Es wird nur die Änderungshistorie von Objekten angezeigt, deren Objektklasse gtraced wird, default = true

Beispiel

```
<Operation ID="ShowHistory" opCode="opShowHistory" onlyIfTraced="true"/>
```

37.46 opShowPage

Die Operation ShowPage aktiviert eine Seite des Pages-Applikationselement.

Beispiel

```
<Operation ID="ShowPage" opCode="opShowPage" forPages="pages" pageName="page1"/>
```

37.47 opSwitchMinimized

Minimiert bzw. maximiert ein Applikationselement. Bei einem minimierten Applikationselement ist nur noch der Header sichtbar. Das Attribut `forElement` **MUSS** definiert werden.

Beispiel

```
<Operation ID="SwitchTreeMinimized" opCode="opSwitchMinimized" forElement="tree1"/>
```

37.48 opUnbind

Die Operation Unbind trennt die Source-Objekte von den Target-Objekten für die angegebene Assoziation.

Die Operation kann in Menüs, PopupMenüs und im Drag&Drop verwendet werden.

Das `unbindAll`–Attribut gibt an, ob ein Unbind oder AnbindAll ausgeführt wird:

Wenn `unbindAll="false"` ist, wird ein Unbind ausgeführt: `Source.Unbind(assoc, Target)`
wenn `unbindAll="true"` ist, wird ein UnbindAll ausgeführt: `Source.UnbindAll(assoc)`

Das `direction`–Attribut gibt die Richtung für ein Unbind oder UnbindAll an:

Wenn `direction="source-target"` ist (Default), wird ein Unbind oder UnbindAll von Source zum Target ausgeführt: `Source.Unbind(assoc, Target) oder Source.UnbindAll(assoc)`
wenn `direction="target-source"` ist, wird ein Unbind oder UnbindAll vom Target zur Source ausgeführt: `Target.Unbind(assoc, Source) oder Target.UnbindAll(assoc)`

Als Eingangsobjekte (Input) für die `Source`-Filternavigation wird die Selektion des Application Elements oder die Drag-Objekte verwendet.

Als Eingangsobjekte (Input) für die `Target`-Filternavigation werden die Objekte in der Zwischenablage oder das Drop-Objekt verwendet.

Beispiel

```
<Operation ID="RemoveFromMyList" opCode="opUnbind"
           assoc="bisB_ObjectsToList" onElement="myList">
    <Source>CLASS Root</Source>
    <Target>START USER VIA bisB_UserToList</Target>
</Operation>
```

37.49 opUndo

Die Operation Undo macht die letzte Änderung in einem Benutzersteuerelement (Editor) rückgänig. Daten, welche zuvor gespeichert wurden, werden nicht rückgängig gemacht. D.h. es ist kein Undo auf Datenbankebene, sondern auf Memoryebene.

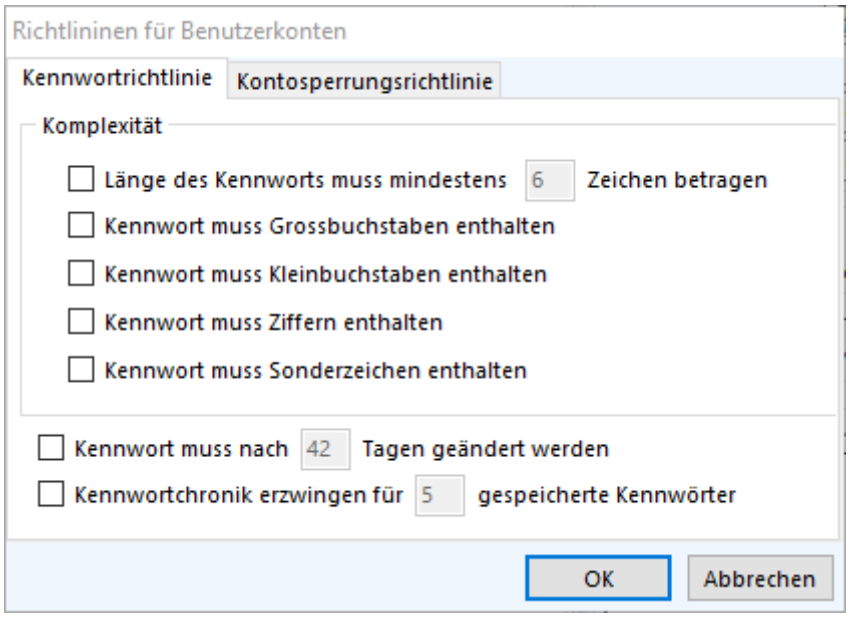
Die Operation kann nur in Menüs und PopupMenüs verwendet werden. Sie unterstützt <Condition> **nicht**.

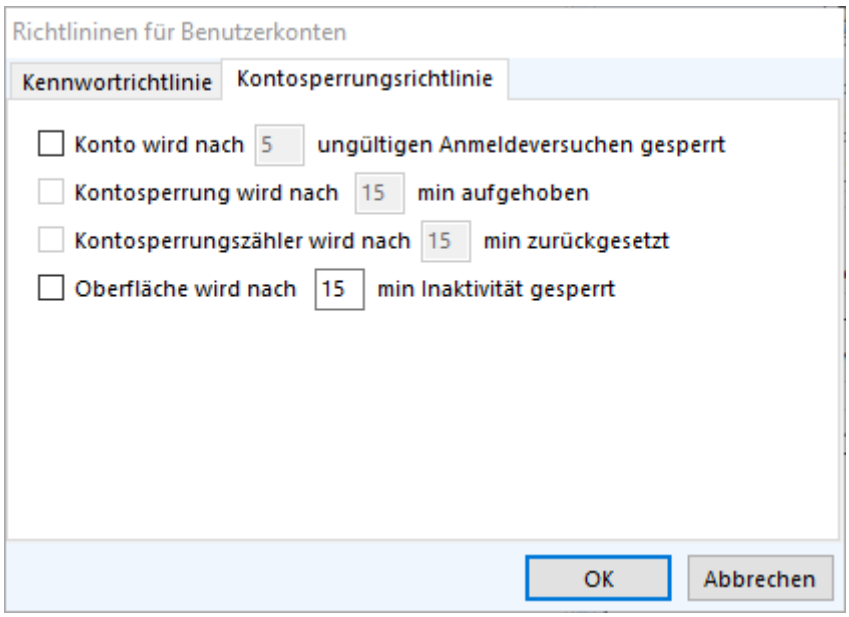
Beispiel

```
<Operation ID="Undo" opCode="opUndo"/>
```

37.50 opUserAccountPolicies

Öffnet ein Fenster zur Bearbeitung der Richtlinien für Benutzerkonten.



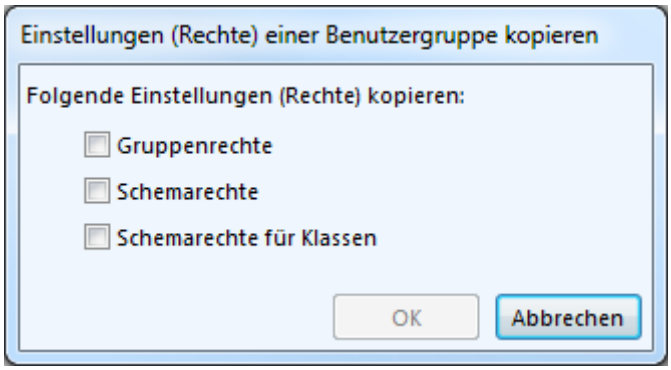


37.51 opUserAccountLockReset

Setzt eine Kontosperrung zurück.

37.52 opUserGroupSettingsCopyFrom (erst ab Byron/BIS v4.11.8)

Kopiert die Einstellungen (Rechte) von der im Menü ausgewählten Benutzergruppe auf die Benutzergruppe der Selektion. Die zu kopierenden Rechte werden in einem Folgedialog ausgewählt.



N.B:

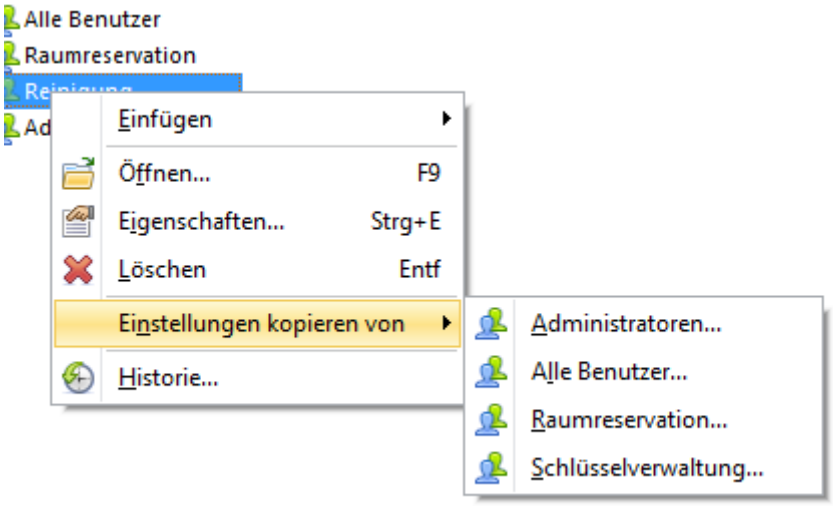
- die gewählten Rechte werden an der Ziel-Benutzergruppe **überschrieben**.
- die Operation unterstützt <Condition> **nicht**.

Das Beispiel

```
      <Operation ID=”UserGroupSettingsCopyFrom” opCode=”opUserGroupSettingsCopyFrom” />
…
    <PopupMenu>
      <Item><Execute name=”UserGroupSettingsCopyFrom”/></Item>
```

```
</PopupMenu>
```

Erzeugt ein Menü mit Auswahl-Submenü:



37.53 opUserPassword

Öffnet das Kennwort ändern-Formular für den selektierten Benutzer. Die Operation unterstützt <Condition> **nicht**.

Beispiel

```
<Operation ID="UserPassword" opCode="opUserPassword" />
```

37.54 opUserSettings

Öffnet das Benutzereinstellungs-Fenster für den selektierten Benutzer. Die Operation unterstützt <Condition> **nicht**.

Beispiel

```
<Operation ID="UserSettings" opCode="opUserSettings"/>
```

37.55 opUserSettingsCopyFrom

Kopiert die Benutzereinstellungen vom im Menü ausgewählten Benutzer auf die Benutzer der Selektion (1:n). Die Operation unterstützt <Condition> **nicht**.

Attribute:

dialogBasedMode Wenn `dialogBasedMode="true"`, wird ein Formular für die Auswahl der Benutzer geöffnet. (Geschieht auch wenn die Anzahl der Benutzer grösser 50 ist.) Vorgabewert für `dialogBasedMode` ist "false". Erst ab Byron/BIS v5.4.7

copyReplacing Wenn `copyReplacing="true"`, werden die Verknüpfungen **ersetzend** kopiert. Wenn `copyReplacing="false"`, werden die Verknüpfungen **additiv** kopiert. Vorgabewert für `copyReplacing` ist "true". Erst ab Byron/BIS v5.4.7

Beispiel

```
<Operation ID="UserSettingsCopyFrom" opCode="opUserSettingsCopyFrom" />
```

37.56 opUserSettingsCopyTo

Kopiert die Benutzereinstellungen vom selektierten Benutzer auf den im Menü ausgewählten Benutzer (1:1). Die Operation unterstützt <Condition> **nicht**.

Attribute:

dialogBasedMode Wenn `dialogBasedMode="true"`, wird ein Formular für die Auswahl der Benutzer geöffnet. (Geschieht auch wenn die Anzahl der Benutzer grösser 50 ist.) Vorgabewert für `dialogBasedMode` ist "false". Erst ab Byron/BIS v5.4.7

copyReplacing Wenn `copyReplacing="true"`, werden die Verknüpfungen **ersetzend** kopiert. Wenn `copyReplacing="false"`, werden die Verknüpfungen **additiv** kopiert. Vorgabewert für `copyReplacing` ist "true". Erst ab Byron/BIS v5.4.7

Beispiel

```
<Operation ID="UserSettingsCopyTo" opCode="opUserSettingsCopyTo" />
```

37.57 opVisibleOnOff

Schaltet die Sichrbarkeit (sichtbar/unsichtbar) des dem Applicationelement `forElement` zugehörigen Layouts um.

Die Sichtbarkeit kann nur verändert werden, wenn alle darüberliegenden Layouts sichtbar sind.

Attribute:

andUp Bei Angabe von `andUp="n"` wird das n-te Parentlayout des entsprechenden Layouts verwendet. Vorgabewert für `andUp` ist "0".

visible Bestimmt, ob das Element sichtbar oder unsichtbar geschalten wird. Ohne Angabe wird hin- und hergeschalten ("toggle"). Erst ab Byron/BIS v4.8.0.

Beispiel

```
<Operation ID="SwitchTreeVisible" opCode="opVisibleOnOff" forElement="tree"
          andUp="0" caption="@sTree@"/>
```

Warnung

Ist das Attribut `andUp` grösser als "0" gesetzt an Applikationselementen, die auch mit einem "X" geschlossen werden können (i.e. optional="true"), dann lässt sich das Applikationselement nach dem Schliessen mit "X" nicht mehr sichtbar machen, da die Operation opVisibleOnOff das falsche Layout sichtbar/unsichtbar schaltet.

37.58 opWorkStateBack

Geht einen Stand in der Anwendungshistorie (Historie der Application Work States) zurück. Ein Stand der Anwendungshistorie kann mit der Operation opWorkStateRecord aufgenommen werden. Ein Stand der Anwendungshistorie wird automatisch aufgenommen, wenn storeWorkStateAfterPropagate der Application oder eines ApplicationElements gesetzt ist.

Die Operation unterstützt <Condition> **nicht**.

Erst ab Byron/BIS v4.11.0.

Beispiel

```
<Operation ID="Back" opCode="opWorkStateBack" />
```

37.59 opWorkStateForward

Geht einen Stand in der Anwendungshistorie (Historie der Application Work States) vorwärts. Ein Stand der Anwendungshistorie kann mit der Operation opWorkStateRecord aufgenommen werden. Ein Stand der Anwendungshistorie wird automatisch aufgenommen, wenn storeWorkStateAfterPropagate der Application oder eines ApplicationElements gesetzt ist.

Die Operation unterstützt <Condition> **nicht**.

Erst ab Byron/BIS v4.11.0.

Beispiel

```
<Operation ID="Forward" opCode="opWorkStateForward" />
```

37.60 opWorkStateRecord

Nimmt einen Stand in der Anwendungshistorie (Historie der Application Work States) auf. Ein Stand der Anwendungshistorie wird automatisch aufgenommen, wenn storeWorkStateAfterPropagate der Application oder eines ApplicationElements gesetzt ist.

Die Operation unterstützt <Condition> **nicht**.

Erst ab Byron/BIS v4.11.0.

Beispiel

```
<Operation ID="Record" opCode="opWorkStateRecord" />
```

N.B.: Wird diese Operation aus einem Formular aufgerufen, dann ist darauf zu achten, dass zuerst `opWorkStateRecord` aufgerufen wird und danach der Zustand des Formulars verändert wird.
Siehe auch Navigation Area des Function Controls.

Ein Beispiel aus einem einfachen Formular in dem ein Geschoss ausgewählt und in einer Grafikfläche angezeigt warden kann. Da ein FunctionControl vorhanden ist, funktionieren die Navigationen (Vorwärts, Zurück).

```
procedure btnShowGraphicClick(Sender: TObject);
begin
  if lbxFloor.DbObject.Count > 0 then begin
    // In Graphic darstellen
    FormInformation.PropagateData('Graphic_Load', 0, '', lbxFloor.DbObject);

    // neuen State aufnehmen
    FormInformation.ExecuteToolOperation('Record');

    dsFloor.DbObject := lbxFloor.DbObject;
    // erst jetzt merken für Speicherung Extension!
  end; //if
end;

procedure FormInformationStoreWorkstate(storage: TStrings);
var
  img : string;
begin
  if dsFloor.DbObject.Count = 1 then begin
```

```
    img := dsFloor.DbObject.GetAttribute('bisB_ObjectImage');
  end else begin
    img := '-';
  end;
  storage.Add(img);
  LogMessage(eltWarning, 'STORE: ' + img);
end;

procedure FormInformationLoadWorkstate(storage: TStrings);
var
  img : string;
begin
  img := storage[0];
  LogMessage(eltWarning, 'LOAD: ' + img);
  if img <> '-' then begin
    dsFloor.DbObject := BisDataSource.DbObject.FilterNavigate('START "' + img +
'"');
    lbxFloor.DbObject := dsFloor.DbObject;
  end; //if
end;
```

37.61 opWorkStateReload

Lädt den aktuell aufgenommenen Stand der Anwendungshistorie (Historie der Application Work States) erneut. Ein Stand der Anwendungshistorie wird automatisch aufgenommen, wenn storeWorkStateAfterPropagate der Application oder eines ApplicationElements gesetzt ist.

`opWorkStateReload` wird verwendet um beispielsweise eine modale Application zu beenden.

Die Operation unterstützt <Condition> **nicht**.

Erst ab Byron/BIS v4.11.8.

Beispiel

```
<Operation ID="WorkStateReload" opCode="opWorkStateReload" />
```

38 Menu / PopupMenu

Da die Definition von Menus und Popupmenüs im Wesentlichen identisch ist, wird die Dokumentation hier zusammengefasst. Popupmenüs können zusätzlich zu Application Elements definiert werden.

Die Konfiguration der Sichtbarkeiten geschieht analog der Sichtbarkeit von Applications, siehe Visibility.

38.1 Attribute von Menu

- `caption = <Text>` Bezeichnung für das Menü.
- `groupRight = <Text>` definiert das benötigte Gruppenrecht.
- `visible = <Boolean>` definiert die Sichtbarkeit des Menus.
- `name = <Text>` Identifikation des Menüs. Bestehende Menüs werden über diese Namen identfiziert. Folgende Menünamen sind vordefiniert:

Name	Bezeichnung
MenuDocument	Dokument
mbPageSetup	Drucker einrichten
mbPrintPreview	Seitenansicht
mbPrint	Drucken
mbPreferences	Einstellungen
mbClose	Schliessen
mbExit	Beenden
MenuEdit	Bearbeiten
mbUndo	Rückgängig
mbCut	Ausschneiden
mbCopy	Kopieren
mbPaste	Einfügen
mbClear	Löschen
mbSelectAll	Alles markieren
MenuView	Ansicht
MenuExtras	Extras
mbExtrasObjList	Objektliste
mbExtrasClpList	Zwischenablage
MenuWindow	Fenster
MenuHelp	?
mbHelpTopics	Hilfethemen
mbHelpInfo	Info

38.2 Enthaltene Elemente

- Menu: Für verschachtelte Menüs.
- Item: Menüeinträge siehe Item

38.3 PopupMenu

38.3.1 Enthaltene Elemente

- Menu: Für verschachtelte Menüs.
- Item: Menüeinträge siehe Item

38.4 Item

38.4.1 Attribute von Item

- `caption = <Text>` definiert die Bezeichung des Menüeintrags.
  Default ist die `caption` der Operation.
  Mit "-" kann eine Trennlinie definiert werden.
- `visible = <Boolean>` definiert die Sichtbarkeit des Items.
- `groupRight = <Text>` definiert das benötigte Gruppenrecht.
- `positionAfter = <text>` definiert die Position anhand des Namens des vorherigen Menüitems.
- `positionBefore = <text>` definiert die Position anhend des Namens des folgenden Menüitems. Achtung: funktioniert nur zuverlässig wenn interne Menüs refrenziert werden. Vgl. die Liste oben.

38.4.2 Enthaltene Elemente

- Execute: Siehe Execute

38.5 Execute

Hinweis: Bei Verwendung von mehreren Execute-Elementen, werden alle Operationen geprüft (Update) und alle gültigen ausgeführt (Execute).

38.5.1 Attribute von Execute

`name = <Text>` definiert die Operations-ID, welche ausgeführt wird. Execute wird in Item, DragDrop und ToolbarButton verwendet.

39 Toolbar

Es sind beliebig viele Toolbars erlaubt. Die Toolbars können im Fenster verschoben werden; die Positionen werden beim Schliessen in der Registry abgelegt und beim Öffnen wiederhergestellt.

Die Konfiguration der Sichtbarkeiten von Toolbars und Toolbarelementen geschieht analog der Sichtbarkeit von Applications, siehe Visibility.

Beispiel:

```
<Toolbar caption="Bearbeiten">
    <ToolbarButton><Execute name="Undo"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="CutClipboard"/></ToolbarButton>
    <ToolbarButton><Execute name="CopyClipboard"/></ToolbarButton>
    <ToolbarButton><Execute name="Paste"/></ToolbarButton>
    <ToolbarButton><Execute name="Delete"/></ToolbarButton>
    <ToolbarSeparator/>
    <ToolbarButton><Execute name="TreeSearch"/></ToolbarButton>
    <ToolbarSeparator/>
    <Menu icon="iconView" caption="Ansicht" hint="Ansichten bearbeiten, etc.">
        <Item><Execute name="ViewEdit"/></Item>
        <Item><Execute name="ViewEdit"/></Item>
        <Item><Execute name="ViewNew"/></Item>
        <Item><Execute name="ViewDelete"/></Item>
    </Menu>
</Toolbar>
```

39.1 Attribute von Toolbar

- `caption = <Text>` definiert die (optionale) Bezeichung der Toolbar. Nur Toolbars mit Bezeichnungen können vom Benutzer im Kontextmenü ausgeblendet werden.

39.2 Enthaltene Elemente

- ToolbarButton: Für einen Button, um eine Operation auszulösen
- Menu: Für ein Menü (erst ab Byron/BIS v4.4.6)
- ToolbarSeparator: Für eine vertikale Trennlinie in der Toolbar.

39.3 ToolbarButton

39.3.1 Attribute von ToolbarButton

- `caption = <Text>` definiert die Bezeichung des Buttons.
  Default ist die `caption` der Operation.
  Der Button wird nur textuell beschriftet, wenn kein Icon vorhanden ist. (siehe auch `forceShowCaption`-Attribut).
- `forceShowCaption = <Boolean>` zeigt die Caption auch an, wenn ein Icon vorhanden ist
  Default ist `false`
  Ab Byron/BIS v4.11.1.

- `hint = <Text>` definiert den Hint des Buttons.
  Default ist der `hint` der Operation.
- `icon = <Text>  definiert das zu verwendene Icon.`
  `Default ist das icon der Operation.`
  Ab Byron/BIS v4.11.8: dem Icon können mit ‚+' Overlays hinzugefügt werden.
  Z.B. `icon = "bisP_Order+state_new"`
- `groupRight = <Text>` definiert das benötigte Gruppenrecht.

39.3.2 Enthaltene Elemente

- Execute: Siehe Execute

39.4 ToolbarSeparator

Für das Element ToolbarSeparator sind keine Attribute und enthaltene Elemente erlaubt.

40 DragDrop

40.1 Attribute von DragDrop

- `keyState = <Text>` definiert, für welche Tastenkombination die Drag&Drop-Operation ausgeführt werden soll.
  Als Default gilt das Gegenteil des übergeordneten Layout, bzw. horizontal beim Obersten.

40.2 Verwendung/Interpretation der KeyStates in Drag&Drop

Während einer `<DragDrop>`-Operation mit der linken Maustaste können verschiedene Tasten gedrückt werden, mit welcher die gewünschte Operation ausgewählt wird. Die Operation wird mit Hilfe des `keyState`-Attributs einer Tastenkombination zugeordnet. Es existieren folgende KeyStates:

- "-" keine Taste gedrückt
- "Ctrl" Ctrl bzw. Strg gedrückt
- "Shift" Shift bzw. Umsch gedrückt
- "Alt" Alt gedrückt
- "Ctrl+Shift" Ctrl und Shift gedrückt
- "Ctrl+Alt" Ctrl und Alt gedrückt
- "Shift+Alt" Shift und Alt gedrückt
- "Ctrl+Shift+Alt" Ctrl und Shift und Alt gedrückt

Ist kein `keyState` definiert, dann kann die Operation **nicht** mit der linken Maustaste gestartet werden (sondern nur mit dem Kontextmenü der rechten Maustaste).

Sind bei einer `<DragDrop>`-Operation mit der **linken** Maustaste mehrere Operationen möglich, dann wird ebenfalls ein Kontextmenü angezeigt.

Im Kontextmenü, welches bei einer `<DragDrop>`-Operation mit der rechten Maustaste angezeigt wird, sind Einträge derjenigen Operationen **fett**, die mit der aktuellen Tastenkombination bei Verwendung der linken Maustaste gestartet würden.

`cursor = "show" ¦ "move" ¦ "copy" ¦ "link"` definiert den Mauscursor für das Drag&Drop.

Als Default wird folgende Zuordnung verwendet:

- "show" wenn keine Taste gedrückt wird
- "move" wenn Shift bzw. Umsch gedrückt wird
- "copy" wenn Ctrl bzw. Strg gedrückt wird
- "link" wenn Ctrl und Shift gedrückt wird

`caption = <Text>` definiert die Bezeichung der Operation für das Popupmenü.

Default ist die `caption` der Operation.

40.3 Enthaltene Elemente

- Execute: Siehe Execute

Beispiel

```
<DragDrops>
    <DragDrop caption="Zeigen" keyState="-">
        <Execute name="actShow"/>
    </DragDrop>
    <DragDrop keyState="Ctrl"><Execute name="CopyForm"/></DragDrop>
    <DragDrop keyState="Ctrl"><Execute name="CopyMenu"/></DragDrop>
    <DragDrop keyState="Shift"><Execute name="MoveForm"/></DragDrop>
    <DragDrop keyState="Shift"><Execute name="MoveMenu"/></DragDrop>
    <DragDrop keyState="-"><Execute name="BindActionToMenu"/></DragDrop>
    <DragDrop keyState="-"><Execute name="BindMenuToAction"/></DragDrop>
</DragDrops>
```

41 Print

Veraltet, bitte ab Byron/BIS Version 4.10.3 die Operation opPrint verwenden!

41.1 Attribute von Print

`appElement = <Text>` verweist auf die ID des Application-Elements, für welches die Seitenansicht aufgerufen wird.
Wird bisher nur von dem Application-Element Grid und Gantt unterstützt.

41.2 Enthaltene Elemente

- PageDecoration: Enthält die Einstellungen für die Seitenansicht im alten Byron/BIS-Format:

Informationen zur PageDecoration

- Es gibt folgende Typen von Feldern: "Text", "Picture", "Remark", "Database", "DbPicture".
- Ein Feld kann folgende Eigenschaften enthalten: PagePos, Anchor, Value, Font, FontSize, Align (right left, center), X, Y, W, H
- Die Einheit von X, Y, W, H ist Millimeter.
- Innerhalb von Value werden folgende Texte erkannt: "$Date$", "$Time$", "$PageNr$", "$TotalPages$" und "\n" für newline.
  ab Byron/BIS Version 4.10.3 stehen zusätzlich "$ApplicationCaption$" für Applikationsüberschrift und "$ApplicationElementCaption$" für das Applikationselementüberschrift zur Verfügung.
- Picture und DbPicture kommen mit Umgebungsvariablen zurecht.
- Für "`object`" in den Feldtypen Database und DbPicture kann "`user`", "`tool`" oder "`container`" eingesetzt werden.
- In "`FilterNav`" kann für die Feldtypen Database und DbPicture eine FilterNavigation angegeben werden. Dabei wird das Datenbankobjekt aus "`object`" als Eingangsmenge für die FilterNavigation verwendet. Das Ergebnis der Filternavigation wird anschliessend für die Auswertung von "`value`" verwendet. (ab Byron/BIS Version 4.10.3)

41.3 Beispiel

```
<Print appElement="grid">
  <PageDecoration>
    Orientation = landscape
    Font        = "Tahoma"
    FontSize    = 10

    Margins
      left   = 20
      right  = 20
      top    = 20
      bottom = 15
    end

    Field:Database
      PagePos   = left, top
      object    = user
```

```
      FilterNav = "VIA bisB_UserToPerson"
      value     = "$Object_Display_Kontext$"
      Y         = -1
      end

    Field:Database
      PagePos = center, top
      object  = container
      value   = "$Object_Display_Kontext$"
      Y       = -1
    end

    Field:DbPicture
      PagePos = right, top
      object  = user
      value   = "$bisB_UserToPerson%bisB_PersonPhoto$"
      Y       = -1
    end

    Field:Picture
      PagePos = left, top
      value   = "%COMPANY_LOGOS_PATH%\Logo.bmp"
      Y       = -1
    end

    Field:text
      pagepos = left, bottom
      value   = "Datum: $Date$"
    end

    Field:text
      pagepos = right, bottom
      Value   = "Seite $PageNr$ von $TotalPages$"
    end
  </PageDecoration>
</Print>
```

Achtung: Globale Parameter erst ab v4.11.8!