Dashboard widgets

URL: https://docs.grassfish.com/docs/dashboard-widgets

Archiviert am: 2025-07-17 18:36:20

Dashboard widgets are supported from the global version 11.0 onwards. They serve the purpose of showing relevant information pertaining to the user in the Grassfish IXM One in the Dashboard tab. Dashboard widgets are based on the gfDashboardWidgetBase which uses several functions of the gfWizardBase. The content of the ascData.json file of Dashboard widgets is similar to the ascData.json file of Wizard spots but differs in the available elements and options which can be used. The reason for the technical relation to the gfWizardBase is to ensure that they can be used in later version in the HTML Composer and transferred to the player.

A zipped dashboard widget is uploaded to the Grassfish IXM One, whereas the file extension must be .dash.zip.

This ZIP file must contain the following files at top level:

- **index.html**: the index.html file is the starting point to the widget. The relative access to additional resources is possible from here, as is common with websites. It is however, not allowed to carry out URL-forwards to other websites on root-level via the ZIP.
- **ascData.json**: the ascData.json file contains all elements in IXM One which can be edited for the widget. The structure and its possible components are described in the following chapter.

If one of these two files are missing, it is not possible to play the widget back correctly. Apart from these two files, the gfWizardBase.js and gfDashboardWidgetBase.js files must be used and instantiated in the root-scope of the spot, whereas the gfWizardBase is handed over to gfDashboardWidgetBase.js. These files can be stored in a subfolder as well. These are the minimum requirements for the Dashboard widget.

API versions

The API version is the direct reference to the corresponding IXM One version. It allows you to define the required functionality of the widget as well as ensure compatibility with IXM One version.

The API version is defined in the ascData.json file via the ApiVersion attribute.

API version
1
2
3

For example, if you try to play back a widget with API version 2 in a IXM One with version 11.0.0, a warning will be displayed during the upload that possibly required functions will not be available and a correct display of the spot cannot be guaranteed.

ascData.json

The ascData.json file is the interface between the widget and the editor. Here it is possible to define elements which can later be edited by users in IXM One. The file also serves the purpose of providing general information on the widget to the Grassfish system, e.g. the possible slot selection which is intended for the widget.

Document

Document – root nodes and widget features	
SpotIdentifier	**Mandatory field, String, ApiVersion: 1** This attribute is assigned by the spot developer and is used to verify the replacement of the widget in IXM One in Example: "MyDemonstrationWidget"
SpotVersion	**Mandatory field, String, ApiVersion: 1** The current version of the widget and JSON structure. Example: "1.0.0"
ApiVersion	**Mandatory field, Int, ApiVersion: 1** Specifies with which systems the widget is compatible with, whereas the API version 1 is valid from 11.0 onwards.
SpotType	**Mandatory field, String, ApiVersion: 1** The spot type must have the value of the Dashboard widget. It serves as an identifier for the backend and must be Fixed value for Dashboard widgets: "SpotType": "DashboardWidget"
NumberOfSlots	**Mandatory field, Int, ApiVersion: 1** Specifies how many slots the widget takes up. For this, valid values range from 1 to 3. Example: "NumberOfSlots": 2
Options	See Options.

Elements	See Elements.

Options

Options - turning features on/off	

PossibleNumbersOfSlots	**Optional, Array, ApiVersion: 1**
	
	The possible number of slots which a widget is allowed to have. These slots should then however b
	
	Examples:
	
	`"PossibleNumbersOfSlots": [ 1, 2, 3 ]`
	`"PossibleNumbersOfSlots": [ 1, 3 ]`

PossibleNumbersOfRows	**Optional, Array, ApiVersion: 2** This defines the line height that a widget occupies in the dashboard. Accordingly, these heights sho Example: `"PossibleNumbersOfRows": [ 1, 2 ]`

```
 "PossibleNumbersOfSlots": [
   1,
   2,
   3
]
"PossibleNumbersOfSlots": [
   1,
   3
]
```

```
 "PossibleNumbersOfRows": [
   1,
   2
]
```

HasDetailView	**Optional, Boolean, ApiVersion: 1**
	
	With this option, it is possible to enable the option in the Grassfish IXM One to open a detail view fo
	the slot size:
	
	Example:
	
	"HasDetailView": true
	
	Example implementation:
	
	` var detail;`
	
	`// ApiVersion 1`
	`detail = !!GFSpotBase.findUrlParam('detailMode');`
	
	`// ApiVersion 2`
	`detail = gfDashboardWidgetBase.isDetailView();`
	
	`if(detail){`
	`//do something`
	`}`

HasPrintSupport	**Optional, Boolean, ApiVersion: 1** With this option, the printing request action is activated in the Grassfish IXM One. This feature must Example: “HasPrintSupport” : true Example implementation:

```
 gfDashboardWidgetBase.registerPrintRequested(
function(){
var data, file;

data = "Ein beliebieger Text...";
file = "export.csv";
gfDashboardWidgetBase.saveToFile(data, file);
});

…

function onPrintRequested()
{
  var data = "Entity" + splitChar + "Name" + splitChar + "Id" + splitC
  var values;
  var item;
  for(var k in multiValues)
  {
    if(multiValues.hasOwnProperty(k))
    {
      values = multiValues[k];
      if(values)
      {
        for(var i = 0; i < values.length; i++)
        {
          item = values[i];
          data += k + splitChar + item.Name + splitChar + item.Id;
          if(item.BoxId)
          {
            data += splitChar + item.BoxId;
          }
          data += "\n";
        }
      }
    }
  }
  gfDashboardExtension.saveToFile(data, "EntityData.csv");
}
```

UseSingleElementUpdate

Optional, Boolean, ApiVersion: 1

With this option, it is possible to specifically react to changes in the respective element and it is not

If this option is enabled, you do not have to necessarily register an Eventhandler for each element
changed.

Example:

"UseSingleElementUpdate": true

Example implementation:

```
 gfDashboardWidgetBase.registerPrintRequested(
function(){
var data, file;

data = "Ein beliebieger Text...";
file = "export.csv";
gfDashboardWidgetBase.saveToFile(data, file);
});

…

function onPrintRequested()
{
  var data = "Entity" + splitChar + "Name" + splitChar + "Id" + splitC
  var values;
  var item;
  for(var k in multiValues)
  {
    if(multiValues.hasOwnProperty(k))
    {
      values = multiValues[k];
      if(values)
      {
        for(var i = 0; i < values.length; i++)
        {
          item = values[i];
          data += k + splitChar + item.Name + splitChar + item.Id;
          if(item.BoxId)
          {
            data += splitChar + item.BoxId;
          }
          data += "\n";
        }
      }
    }
  }
  gfDashboardExtension.saveToFile(data, "EntityData.csv");
}
```

DynamicResolution	**Optional, Object, ApiVersion: 2**
	
	With this option you can manage that the wizard spot has a height and width which can be edited by
	
	Example:
	
	`"DynamicResolution": {`
	`  "Active": true,`
	`  "MinWidth": 400,`
	`  "MaxWidth": 2000,`
	`  "MinHeight": 300,`
	`  "MaxHeight": 3000`
	`}`

Elements

Checkbox

Checkbox	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "boolean"
Value	**Mandatory field, Boolean, ApiVersion: 1** Example: "Value": false
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="]

Number selection

Number

Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "number"
Value	**Mandatory field, Number, ApiVersion: 1** Example: "Value": 42
Options -> Minimum	**Optional, Number, ApiVersion: 1** Specifies the minimum value of the number selection. Example: "Minimum": 1.0
Options -> Maximum	**Optional, Number, ApiVersion: 1** Specifies the maximum value of the number selection. Example: "Maximum": 999
Options -> StepSize	**Optional, Number, ApiVersion: 1** Specifies the rounding value of the number selection. Example: "StepSize": 0.1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|">"|">="|"<"|"<="]

Color selection

Colorpicker

Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "color"

Value	**Mandatory field, String, ApiVersion: 1**
	
	The format depends on the UseRGBA option.
	
	Example:
	
	`UseRGBA !== true :"Value": "#FF0000"`
	
	`UseRGBA === true : “Value”: “rgba(72,70,189,0.64)`
	
	If the value is color value in the RGB hex format, starting with a hash. Since the global ve
	
	Example:
	
	"Value": "#FF0000"

Options -> UseRGBA	**Optional, Boolean, ApiVersion: 1** By means of this option, the advanced color picker can be activated, which also supports Example: "UseRGBA": true
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Date

Date	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table

DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "date"
Value	**Mandatory field, String, ApiVersion: 1** A date string in the format YYYY-MM-DDThh:mm:ss. Example: "Value": "2015-09-13T00:00:00"
Options -> MinDate	**Optional, String, ApiVersion: 1** By means of this option the minimum date can be specified. The entry may not go belo Example: "MinDate": "2015-09-11T00:00:00"
Options -> MaxDate	**Optional, String, ApiVersion: 1** By means of this option, the maximum date can be specified. The entry may not excee Example: "MaxDate": "2015-09-15T00:00:00"
Options, Keyword ‘Today’	**Optional, String, ApiVersion:1** For the threshold values MinDate and MaxDate, it is also possible to assign the keywo Example: „MaxDate“: „Today“
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|">"|">="|"<"|"<="]

Date Range

Date	
Id, DisplayName	**Mandatory fields, ApiVersion: 2** See Elements.

DataType	**Mandatory field, String, ApiVersion: 2** "DataType": "dateRange"
Value	**Mandatory field, Object, ApiVersion: 2** The format depends on the options TimeEnabled and TimeZoneEnabled. Example: Default: `"Value": { "From": "2015-09-13T00:00:00", "To": "2015-09-13T00:00:00" }` TimeEnabled === true: `"Value": { "From": "2015-09-13T11:15:00", "To": "2015-09-13T23:30:00" }` TimeEnabled === true && TimeZoneEnabled === true: "Value": { `"Value": { "From": "2015-09-13T11:15:00Z", "To": "2015-09-13T23:30:00Z" }`
Options -> MinDate	**Optional, String, ApiVersion: 2** By means of this option it is possible to set the minimum date. The entry may not go b Example: "MinDate": "2015-09-11T00:00:00"
Options -> MaxDate	**Optional, String, ApiVersion: 2** By means of this option it is possible to set the maximum date. The entry may not exc Example: "MaxDate": "2015-09-15T00:00:00"

Options -> TimeEnabled	**Optional, String, ApiVersion: 2** By means of this option the time entry can be activated. Example: "TimeEnabled": true
Options -> TimeZoneEnabled	**Optional, String, ApiVersion: 2** By means of this option, it is possible to activate the entry of the time zone. Example: "TimeZoneEnabled": true **Note:** For this option, TimeEnabled as well as the configuration of IXM One for the sup
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": "?"
Options, Schlüsselwort "Today"	**Optional, String, ApiVersion: 2** For the threshold values MinDate and MaxDate it is also possible to define the keywor Example: "MaxDate": "Today"

Time

Time	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "time"

Value	**Mandatory field, String, ApiVersion: 1**
	The format depends on the option WithSeconds.
	Example:
	`WithSeconds !== true: "Value": "13:57"`
	`WithSeconds === true: "Value": "13:59:57"`

Options -> WithSeconds	**Optional, Boolean, ApiVersion: 1** By means of this option, the entry of seconds can be enabled. Depending on the setti Example: "WithSeconds": true
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|">"|">="|"<"|"<="]

Dropdowns

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "list"

Items	**Mandatory field, Array**
	Example:
	`"Items": [`
	`"Cow",`
	`"Horse",`
	`"Sloth"`
	`]`

Value	**Mandatory field, String, ApiVersion: 1** The selected entry from the items array is saved under ‚value‘ (in the non-translated v Example: "Value": "Horse"

Translations	**Optional, Object, ApiVersion: 1**
	
	For dropdowns, it is possible to define translations for every value in Translations.
	
	Example:
	
	` "Translations": {`
	`"de": {`
	`"DisplayName": "Irgendein Listeintrag2",`
	`"Cow": "Kuh",`
	`"Horse": "Pferd",`
	`"Sloth": "Faultier"`
	`}`
	`}`

Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Radio button

Radio button	
Id, DisplayName	**Mandatory fields, ApiVersion: 2** See Elements.
DataType	**Mandatory field, String, ApiVersion: 2** "DataType": "radiobutton"

Items	**Mandatory field, String[], ApiVersion: 2**
	
	Example:
	
	`"Items": [`
	`    "Cow",`
	`    "Horse",`
	`    "Sloth"`
	`]`

Value	**Mandatory field, String, ApiVersion: 2** The selected entry from the items array is saved under ‚value‘ (in the non-translated Example: "Value": "Horse"

Translations	**Optional, Object, ApiVersion: 2**
	
	In dropdowns, it is optional to define translations for each value. For this, the key cor
	
	Example:
	
	`"Translations": {`
	`"de": {`
	`"DisplayName": "Tierauswahl",`
	`"Cow": "Kuh",`
	`"Horse": "Pferd",`
	`"Sloth": "Faultier"`
	`}`
	`}`

Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Simple text input

SimpleText	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table

DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "simpleText"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects, each have a text property where an unformatted text is
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Text": "some text."`
	`        },`
	`        {`
	`                "Text": "Another text."`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter you can control how many instances of a text i Example: "NumberOfInstances": 1
Options -> MultilineSize	**Optional, Int, ApiVersion: 1** By means of this parameter you can control how large the text input is displ Example: "MultilineSize": 4
Options -> MaxChars	**Optional, Int, ApiVersion: 1** By means of this parameter you can determine how many characters the u Example: "MaxChars": 120
Options -> Required	**Optional, Boolean, ApiVersion: 1** By means of this parameter you can ensure that something has been enter Example: "Required": true

Options -> ValidationRegExp	**Optional, String, ApiVersion: 1** By means of this parameter you can store a regular expression that must b Example: "ValidationRegExp": "[\\d]*"
Options -> ValidationRegExpDescription	**Optional, String, ApiVersion: 1** By means of this parameter you can define the error message for the regul Example: "ValidationRegExpDescription": "Hier sind nur Zahlen erlaubt."
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Spot group selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "spotGroup"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects which each have an ID and the name as properties (at
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Id": 123,`
	`                "Name": "My spot group"`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter you can control how many instances of a spot Example: "NumberOfInstances": 1
Options -> IncludeTemplates	**Optional, Boolean, ApiVersion: 1** This parameter specifies whether it is possible to select template spot grou Example: "IncludeTemplates": true
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Media group selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "mediaGroup"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects which each have an ID and the name (at the time of th
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Id": 123,`
	`                "Name": "My media group"`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control how many instances of a me Example: "NumberOfInstances": 1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Location selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "location"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects which each have an ID and the name (at the time of t
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Id": 123,`
	`                "Name": "My location"`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control the number of instances of a Example: "NumberOfInstances": 1

Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Player selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "player"
Value	**Mandatory field, Array, ApiVersion: 1** An array of objects, which each have an ID and the name (at the time Example: `"Value": [ { "Id": 123, "Name": "My player" } ]`
Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control how many instances of a Example: "NumberOfInstances": 1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Category selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "category"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects, which each have an ID and the r
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Id": 123,`
	`                "Name": "My category`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control how m hierarchical order. Example: "NumberOfInstances": 1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Spot selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.

DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "spot"

Value	**Mandatory field, Array, ApiVersion: 1**
	
	An array of objects, which each have an ID and the name (a
	
	Example:
	
	`"Value": [`
	`        {`
	`                "Id": 123,`
	`                "Name": "Mein Spot"`
	`        }`
	`]`

Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control how many insta Example: "NumberOfInstances": 1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Media selection

List	
Id, DisplayName	**Mandatory fields, ApiVersion: 1** See Elements table.
DataType	**Mandatory field, String, ApiVersion: 1** "DataType": "media"

Value	**Mandatory field, Array, ApiVersion: 1** An array of objects, which each have an ID and the n Example: `"Value": [ { "Id": 123, "Name": "My medium" } ]`
Options -> NumberOfInstances	**Optional, Int, ApiVersion: 1** By means of this parameter, you can control how mai Example: "NumberOfInstances": 1
Options -> Conditions	**Optional, Object, ApiVersion: 2** Details see Conditional Elements. "Operator": ["!="|"=="|"?"]

Edition selection

List
Id, DisplayName
DataType

Value
Options -> NumberOfInstances
Options -> Conditions

Conditional elements - Conditions

Via Conditions you can define an optional number of conditions which can be linked via AND-operations. A condition is defined via the option object and consists of four properties:

- **Action**: defines what happens during a successful evaluation of the verification. Currently only the ‚HIDE‘ action for hiding the entry fields is supported.
- **TargetId**: corresponds to the ID of the element which should be tested.
- **Operator**: defines the type of verification. The availability of the operators depends on the respective type of the element which should be tested. Details can be found in the properties of the respective element.
- **Value**: the value which is to be verified.

Example:

```
{
 "Id": "boolean1",
 ...
 "Id": "radiobutton1",
 ...
 "Id": "optionalInput",
    ...
    "Options": {
      ...
      "Conditions": [
        {
          "Action": "HIDE",
          "TargetId": "boolean1",
          "Operator": "==",
          "Value": true
        },
        {
          "Action": "HIDE",
          "TargetId": "radiobutton1",
          "Operator": "==",
          "Value": "DO_NOT_SHOW"
        }
      ]
    }
  }
```

In the above listed definition, the element ‚optionalInput‘ is checked against two conditions.

1. Condition 1 is met when the element ‚boolean1‘ has the value *true*, i.e., the checkbox is activated in the editor.
2. Condition 2 is met, when the element ‘radiobutton1’ has the value *DO_NOT_SHOW*. The element ‚optionalInput‘ is hidden, when both conditions are met.

GFDashboardWidgetBase

The GFDashboardWidgetBase is an extension of the GFWizardBase. This is a JavaScript Lib which, in addition to the GFWizardBase, must be integrated in Dashboard widgets to ensure proper functionality of the Grassfish system.

The following subchapters describe the events and additional information relating to their use.

Use

The GFDashboardWidgetBase file can be nested in subfolders but it should not be edited. Add the file to the website via the <script> tag following the GFWizardBase.

```
 <script src="gfWizardBase.js"></script>

<script src="gfDashboarWidgetBase.js"></script>
```

It is only possible to access the methods of the GFDashboardWidgetBase, once the JavaScript file has been initialised. For this, use the onload method of the <body> tag:

<body onload="init()">

First, an instance of the GFDashboardWidgetBase must be created in the script. The constructor takes an instance of the GFWizardBase as a parameter. In contrast to the GFSpotBase, the GFDashboardWidgetBase is not a static class and must be instantiated globally.

```
 //gfDashboardWidgetBase must be accessed globally!
var gfDashboardWidgetBase = new GFDashboardWidgetBase(new GFWizardBase());

function init()
{
//register all events
gfDashboardWidgetBase.registerDataChangedHandler(onDataChanged);
//notify the container that you are ready
gfDashboardWidgetBase.sendReady();

alert(gfDashboardWidgetBase.getVersion());
}

function onDataChanged(jsonData)
{

}
```

Initialization

Initially, the spot must send a 'Ready' event so that the containers know that they can now communicate with spot. This invocation looks as follows:

gfDashboardWidgetBase.sendReady();

Not until after the spot has reported itself to be ready does it receive its JSON data from the container. If the data have been set, the spot automatically sends an InitComplete event to the container, which is forwarded to the player. As a result of this the player knows that the spot can be displayed.

You can optionally delay the InitComplete by adding a true parameter to the sendReady. As a result, the InitComplete event is not automatically triggered upon setting the data, but must be manually triggered.

```
 gfDashboardWidgetBase.sendReady(true);

//do your asynchronous stuff, e.g. call a webservice

gfDashboardWidgetBase.sendInitComplete();
```

Note

If you would like to manually trigger InitComplete, but then do not trigger it, errors will occur during playback. If you also use the GFSpotBase in an HTML Wizard spot, please trigger ONLY the sendInitComplete() of the GFWizardBase and not that of the GFSpotBase.

Receiving data

In order that the spot receives the data filled in by the user, he must register for the dataChanged event. This takes place as follows:

gfDashboardWidgetBase.registerDataChangedHandler(onDataChanged);

Function	Registers a call-back function that is initially invoked and if a
Parameters	function(jsonData) jsonData the JSON data that come from the HTML Wizard (

Examples	`gfDashboardWidgetBase.`
	`  registerDataChangedHandler(onDataChange`
	
	`function onDataChanged(jsonData)`
	`{`
	`var elementData = gfDashboardWidgetBase.g`
	`document.getElementById("myElement").inne`
	`}`

Minimum Version	**ApiVersion: 1** Grassfish IXM One: 11.0.0

Starting/Stopping animations

When the Dashboard widget is loaded in the Grassfish IXM One, IXM One sends a play command via call-back in order to start the animations.

gfDashboardWidgetBase.registerPlayHandler(onPlay);

Function	Starts the animation in the Dashboard w
Parameters	None
Examples	gfDashboardWidgetBase. registerPlayHandler(startMyAnimations)
Minimum Version	**ApiVersion: 1** IXM One: 11.0.0

gfDashboardWidgetBase.registerStopHandler(onStop);

Function	This function does not currently have ar

Parameters	None
Example	gfDashboardWidgetBase. registerStopHandler(resetMyAnimations
Minimum Version	**ApiVersion: 1** IXM One: 11.0.0

Detail view

In addition to the three possible resolutions in the Dashboard, it is possible to activate a detail view for the widget as well. Detail mode has a resolution of 1024x576 and serves the purpose of providing the user with a more complex way to display reports (e.g. for reporting) via the widget. Via the isDetailView method, it can be requested if the widgets should be displayed in the detail view.

```
 if(gfDashboardWidgetBase.isDetailView())
{
  //display Widget in detail mode.
}
```

Report View

Furthermore, dashboard add-ons can also be used in the Reporting area of IXM One. Additional space for the presentation of according data has been allocated for improved clarity. The size is not set in this view, as it corresponds to the available space in IXM One. It is thus important to make sure that the display is suitable. The methode isReportView can be queried to see if the widget should be shown in the Report view.

```
 if(gfDashboardWidgetBase.isReportView())
{
  // display widget in report mode
}
```

Printing commands

Via the Grassfish IXM One, the Dashboard widget can be prompted to generate a print preview. This can be a new browser tab or a backend request for the generation of a CSV file. This is carried out via the call-back registerPrintHandler. In the GFDashboardWidgetBase, there is a support function called saveToFile which is described below in this chapter.

gfDashboardWidgetBase.registerPrintHandler(onPrintRequested);

Function	Call-back function to process the print command from IXM One in the Dashboard widget.
Parameters	None

Examples	`gfDashboardWidgetBase.`
	`  registerPrintHandler(onPrintRequested);`
	
	`function onPrintRequested()`
	`{`
	`}`

Minimum Version	**ApiVersion: 1** IXM One: 11.0.0

gfDashboardWidgetBase.saveToFile(value, fileName, filetype);

Function	Support function to open a file dialogue in order to enable the implementation of saving data.
Parameters	Value string, the data *fileName* string, the file name *filetype* string, the type of file, optional, default: "application/octet-stream"

Examples	`var content;`
	
	`content = "ID,TEXT"\n";`
	`content += "1,Ich"\n";`
	`content += "2,werde"\n";`
	`content += "3,exportiert"\n";`
	
	`gfDashboardWidgetBase.saveToFile(content, "export.csv", "text/csv"`

Minimum Version	**ApiVersion: 1** IXM One: 11.0.0

Logging

It is possible to display on-screen logging. Messages which should be displayed here must be sent to the container via sendLog. By means of call-back registerLogHandler you can then also connect to these log messages in order to continue to react to them in the spot.

gfDashboardWidgetBase.sendLog("message");

Function	Logs the message in the on-screen log of the ascInterface. The log must
Parameters	function(message) *message* Message which is to be logged
Examples	gfDashboardWidgetBase.sendLog("message");
Minimum version	**ApiVersion: 1** IXM One: 11.0.0

gfDashboardWidgetBase.registerLogHandler(onLog);

Function	Call-back function for sendLog for processing in the Dashboard widget.
Parameters	function(message) message Message which is to be logged

Examples	
	```
	 gfDashboardWidgetBase.registerLogHandler(onLog);

	function onLog(message)
	{
	}
	```

Minimum version	**ApiVersion: 1** IXM One: 11.0.0

Uploading to IXM One

Certain standard Dashboard widgets are automatically uploaded to the system. Custom widgets can be uploaded via the administration under **Dashboard > Widgets** to the widget group **Widget templates**. From there it is then possible to drag variants to the corresponding widget groups. Further information on this can be found in the Online Help.

Editing in IXM One

If a dashboard widget should be updated, it is only possible in the administration under **Dashboard > Widgets** to the widget group **Widget templates**. Here the respective widget must be highlighted and then selected under **Tools > Replace widget**. The upload manager is opened, and the new version of the dashboard widget must be dragged here prior to starting the upload. The ID of the uploaded Dashboard widgets must match the ID of the selected Dashboard widget in order to update it.