

Upgrading to Servoy 6.0

In This Chapter

- [Upgrading installations](#)
 - [Upgrading Servoy Application Server](#)
 - [Upgrading Servoy Developer](#)
- [Upgrading existing Solutions](#)
 - [Behavior changes](#)
 - [Changes that might break existing code](#)
 - [Behavior changes](#)
 - [Changes affecting the appearance](#)
 - [Resolving warnings and errors](#)
 - [Improved design-time JavaScript code validation](#)
 - [Deprecated API](#)
 - [Designtime API](#)
 - [Runtime API](#)
 - [Utilizing the new Solution Design features](#)
 - [New features](#)
 - [Windowing API](#)
 - [Drag 'n' Drop](#)
 - [Inheritance](#)
 - [Encapsulation](#)
 - [CSS Row Styling](#)
 - [onRender event](#)
 - [SolutionModel extensions](#)
 - [DatabaseManager extensions](#)
 - [Tablefilter changes](#)
 - [visible & enabled Designtime properties](#)
 - [RuntimeXxxx types](#)
 - [Extended scripting API for RuntimeGroup](#)
 - [Solution deeplink normalization](#)
 - [Extended ClientDesign](#)
 - [Tooltip display settings](#)
 - [i18n additions](#)
 - [HTTP plugin updates](#)
 - [Restful plugin updates](#)
 - [File plugin updates](#)
 - [Maintenance plugin updates](#)
 - [OpenId plugin](#)

Upgrading installations

Upgrading Servoy Application Server

As Servoy 6 is a major release a fresh Servoy Application Server installation is required.

See [Installing the Application Server](#) for more information.

Upgrading Servoy Developer

As Servoy 6 is a major release a fresh Servoy Developer installation is required.

See [Installing Developer](#) for more information.

Upgrading existing Solutions

Upgrading existing Solutions to Servoy 6 is as easy as opening Servoy Developer 6 and getting the Solutions into the workspace. As Solutions that are opened in Servoy Developer 6 are updated automatically to be compatible with Servoy 6, they will stop being compatible with earlier versions of Servoy. It is therefore advised to checkout the solutions into a new workspace or import the solutions into the new workspace.

Once imported, the Solutions can be tested with Servoy 6. Servoy 6 introduces a few minor behavior changes, some of which could potentially break existing code. It should be investigated if the upgraded solutions are affected by these changes. The behavior changes are listed and discussed in the paragraph [Behavior Changes](#).

After taking care of the behavior changes, there are 2 tasks left:

- [Resolving warnings and errors](#)
- [Utilizing the new Solution Design features](#)

Behavior changes

This paragraph describes the behavior changes that are introduced in Servoy 6.0. There are three categories of behavior changes:

1. Changes that might break existing code
2. Real behavior changes
3. Changes that affect the display of the UI

Changes that might break existing code

- Bugfix: JSFoundSet.getRecordIndex(JSRecord) now returns -1 instead of 0 when the record is not found, to be inline with other functions that return -1 to indicate "not found"
- BugFix: When passing JavaScript Arrays into the Scripting API (Servoy Scripting API or plugins), previously empty Array positions were removed, now they are reserved
- plugins.rawSQL.executeStoredProcedure() now returns the last resultset returned by the stored procedure, instead of the first, as this is most likely the resultset to continue with.
- In the solutionModel when asking for a specific element on a JSForm, if the element is not on the JSForm, the super form hierarchy will be inspected and return the element if the element is inherited. For the plural getElement() type of functions, an optional paramter has been added to include the inherited elements in the return value or not.
- When adding a tableFilter on a column that already has a tablefilter on it, the newly applied filter will not override the existing filter, but the new filter will be appended, so both the existing and newly added tableFilter are in effect.
- To implement the old behavior, the TableFilters need to be created with a name and then when applying a new TableFilter the old one needs to be removed through scripting.
- BugFix: A bug was fixed that allowed making elements visible in scripting on which there was a security constraint that made the element invisible. The new behavior is that security takes preference over scripting for "visibility", like was already the case for "enabled"
- The passing of deepLink arguments to the onOpen methods of both the Login solution (if applicable) and the main solution has been normalized:
 - Proper supports for deeplinks that specify multiple values for both the predefined "argument" parameter as well as custom named parameters.
 - The difference between deeplinks using the full syntax ("arguments") or short syntax ("a") for specifying arguments has been made transparent: short syntax will be automatically converted to full syntax.
 - The query arguments in the URL that represent the solutionName or the methodName are filtered out, so only the real arguments are send into the onOpen methodExamples:

```
http://domain:port/servoy-webclient/ss/s/{solutionName}/argument/1/m/{globalMethodName}/a/2/a/3/b/1/b/2/b/3 or
http://domain:port/servoy-webclient/ss?s={solutionName}&a=1&method={globalMethodName}
&a=2&a=3&b=1&b=2&b=3 or
http://domain:port/servoy-client/{solutionName}.jnlp?a=1&a=3&b=1&a=2&b=2&b=3
```

all results in the following arguments applied to the onOpen method

```
arguments[0]: 1
arguments[1]: {argument: [1,2,3], b: [1,2,3]}
```

Behavior changes

- In Smart Clients running under Java 6, Modal Dialogs (JSWindow of type MODAL_DIALOG and the dialogs displayed using the dialogs plugin) will be modal for their parent window (hierargy). This means that when having multiple windows (JSWindow of type WINDOW) open, a Modal dialog opened in one window will only block that window, not the other Windows. This aligns the behavior with the Web Client behavior. Unfortunately, the ability to set the scope of dialog modality in only possible from Java 6 onwards. When running Smart Clients on Java 5 modal dialogs will block all windows.
- The position and dimensions of JSWindow's (Windows and Dialogs) are now persisted over client sessions
- The client that gets started as Authenticator during the login process isn't consuming a license any more
- controller.recreateUI() will keep the existing x and y scrollPosition of the form being recreated.
- application.getTimeStamp() in the Web client will return the timestamp based on the timezone of the Client and the time of the Server: Before it would just return the timestamp based on the time and timezone of the server.
- Tooltips in the Web Client are not shown immediately, but after a short delay, to synchronize the behavior between the Smart and Web Client. Note that through the UIProperties APP_UI_PROPERTY.TOOLTIP_INITIAL_DELAY & APP_UI_PROPERTY.TOOLTIP_DISMISS_DELAY the initial show delay and the dismiss delay are now controllable by the developer in both the Smart and Web Client
- DeepLink arguments are now also applied to the onOpen method of the login solution. As a result of that, the application.getStartupArguments() method is deprecated, as it's no longer needed
- Enhanced Security is now the default mode. The UI in Servoy Developer and in the Application Server to disable it has been removed. It is only possible to disable Enhance Security through setting the relevant property in the servoy.properties file: "servoy.application_server.enhancedSecurity=false"
- All access to server side services like HeadlessClient or the Mail plugin is not allowed anymore from unauthenticated Smart Clients. In order to access serverside services, the user needs to log in first. If the solution requires access to a serverside service prior to the user logging in, the developer needs to expose the services as methods in the Authenticator solution, which can be accesses prior to the user authenticating.

Changes affecting the appearance

- Fonts inside an HTMLArea expressed in "pt" instead of "px" will display slightly smaller due to a bugfix.
- Non-modal Dialogs in the Web Client rendered inside the main window, instead of as new browser window due to JSWindow implementation, allowing multiple modal and non modal in the same window
- Default alignment for text in TableView & Portal headers brought inline (Centered)
- Aligned default border and alignment for all elements between Smart Client and Web Client
- When using labels with labelFor link to other elements in TableViews to control the TableView header rendering, the first label will now determine the height of the header
- Due to added support for vertical and horizontal alignment on labels and buttons in the Web Client, any custom HTML displayed in a non-editable HTMLArea in the Web Client will be offset from the top by 50% of the height of the element, if the vertical alignment is left on Default, with means Center. Set the vertical alignment to Top to get the HTML to display correctly again.
- Due to the added support for an editable HTMLArea in the Web Client, end users will now see an HTML Editor in the Web Client if the HTMLArea is left enabled in the Solution design, whereas in previous versions it would show the raw content of the dataprovider.

Resolving warnings and errors

Servoy 6 features a greatly enhanced build mechanism that checks the validity of the design of a Solution while it is being designed. As a result of this improved build system, existing solutions that get opened in Servoy Developer 6 will most likely contain a lot of Problem markers.

It is important to note that these Problem markers are of the level "warning", meaning that it doesn't necessarily mean something is broken!

The majority of the generated Problem markers are most likely related to the JavaScript code contained in the Solutions, as this is one of the area's where the most improvements were made, but some work is required by the developer to get the best results. More on this is discussed below, in the [Improved design-time JavaScript code validation](#) aline.

Another area where Problem markers are raised that were not raised before is the area of using deprecated scripting API. With every release Servoy improved the capabilities in the scripting API. This sometimes means deprecating parts of the API in favor of new and improved API. Deprecating API means that it is marked as "Should not be used anymore", while it continues to function. In servoy deprecated scripting API is hidden in developer.

As of Servoy 6, Problem markers will be generated on the JavaScript code that accesses deprecated API. The API will still function as before, the Problem markers are just an indication that the used API is marked as "Should not use anymore" and that at one point in time the code ought to be rewritten to utilize the new way of doing things. The [Deprecated API](#) aline below provides an overview of the API that got deprecated in Servoy 6.0 and what the replacement functionality is.

Improved design-time JavaScript code validation

Servoy 6 features greatly improved design-time JavaScript code validation: the entire code base of a solution will be continuously checked for possible coding mistakes, like accessing or setting non existing variables, calling non-existing methods or calling methods with the wrong number or type of parameters.


While this is great once you get going, if you're upgrading an existing solution to Servoy 6, you will most likely get a lot of warnings on your code. Why is that?

JavaScript, the scripting language used in Servoy to write down the business logic is what they call "weak-typed". This means that a variable can hold any type of value, that a function can return any type of value and that the parameter values that can be send into a method call can be of any type. At least: this hold true for JavaScript in it's core.

The new script validation in Servoy Developer 6 however tries to determine if the the properties you access actually exist, if the method you call exist and if you call them with the right number and type of arguments. The validation mechanism tries to be as clever as possible to determine which code is correct and which code isn't, but due to the weak-typed (and dynamic) nature of JavaScript, sometimes clever logic just isn't enough and the validator requires more input from the developer in order to do proper validation. This developer input is done in the form of JSDoc.

While previous versions of Servoy already had support for JSDoc, in Servoy 6 this support has been extended and improved, in order to better facilitate the validation process. For more information on annotating JavaScript using JSDoc see [Annotating JavaScript using JSDoc](#).

While we advise to solve the warnings, so you can also reliable start using some of the other great features of Servoy 6, like automatic code refactoring or search for references, you can ignore the warnings. It's even possible to turn off all the warnings, see Window > Preferences > JavaScript > Error /Warnings.

 Do note that automated refactoring, Search for References or JavaScript Search might not be 100% complete if JavaScript is not properly annotated using JSDoc, due to which the JavaScript code cannot be fully analysed automatically.

Deprecated API

Servoy 6 deprecates parts of the existing API, as it has become obsolete or has been replaced by better alternatives. This section lists the deprecated methods or properties with their replacement where applicable.

Design-time API

The list of deprecated properties in the Design-time API is limited. Except for the deprecation of the "rowBGColorCalculation" on Forms and Portals, the deprecation of the properties require no action on behalf of the developer, as Servoy Developer will automatically handle the changes.

rowBGColorCalculation

The "rowBGColorCalculation" is replaced by both the ability to style odd/even/selected rows using CSS and additionally the new onRender event to do conditional styling. As the calculation or method used for the rowBGColorCalculation property cannot be used for either CSS Styling or the onRender event, the conversion to the new way of doing things is a manual action for the developer.

Forms that have a value set for the rowBGColorCalculation property will continue to show the property in the Form Editor in Servoy Developer and the rowBGColorCalculation functionality will continue to be applied at Runtime. On new forms and existing forms that don't have a value set for the rowBGColorCalculation property, the property will not be visible anymore.

Object	Deprecatd method or property	Replacement	Comment
Form	extendsForm	extends	
	rowBGColorCalculation	CSS Row Styling & onRender event	
Portal	resizeble	resizable	Fix typo
	rowBGColorCalculation	CSS Row Styling & onRender event	
TabPanel	onTabChange	onChange	

Runtime API

The majority of the deprecations in the Runtime API are related to the new windowing API and the refactored HTTP plugin.

Object	Deprecatd method or property	Replacement	Comment
controller	.getContainerName()	controller.getWindow().getName()	
Button Label Fields Rectangle TabPanel SplitPane Portal	.setBorder() .setFont() .getTitleText() .setImageUrl()	.border .font .titleText .imageUrl	Matching getters or setters added, allowing both getting and setting the value. Not all mentioned properties are applicable to each mentioned type of object. Due to a technical issue, Servoy Developer will yield a warning that the old setXxxx/getXxxx is undefined, instead of being deprecated. It will still work however.
RuntimeLabel	.getParameterValue()	N/A	Never worked properly
Date	.setYear()	.getFullYear()	This function is deprecated in the JavaScript specification
Application	.getStartupArguments()	onOpen event handler of the solution	Deeplink arguments are now passed to both the onOpen event handler of the Login and Main solution. The .getStartupArgument() is therefor no longer required to get access to the startupArguments prior to login
	.closeForm()	JSWindow.hide()	Servoy 6 contains a complete new Windowing API. A window or dialog is now an instance of the JSWindow class to which a lot of the methods have been moved. New methods were added to the application and controller objects to interact with windows
	.getWindowHeight()	JSWindow.getHeight()	
	.getWindowWidth()	JSWindow.getWidth()	
	.getWindowX()	JSWindow.getX()	
	.getWindowY()	JSWindow.getY()	
	.setWindowLocation()	JSWindow.setLocation()	
	.setWindowSize()	JSWindow.setSize()	
	.showFormInDialog()	application.createWindow('name',JSWindow.DIALOG).show('formName') application.createWindow('name',JSWindow.MODAL_DIALOG).show('formName')	The method application.createWindow(..) returns an object of type JSWindow, which has an Scripting API to control additional things like the title, resizableity, initial bounds and whether or not to show a textToolbar
	.showFormInWindow()	application.createWindow('name',JSWindow.WINDOW).show('formName')	
UICONSTANTS	.FULL_SCREEN	JSWindow.FULL_SCREEN	
JSForm	.rowBGColorCalculation	CSS Row Styling & onRender event	
JSPortal	.resizeble	.resizable	Fix typo
	.rowBGColorCalculation	CSS Row Styling & onRender event	
JSTabPanel	.onTabChange	.onChange	

database Manager	.getFoundSetDataProviderAsArray()	.convertToDataSet(['dataProviderId']).getColumnAsArray(1)	Duplicate functionality
ServoyException	.INVALID_INPUT_FORMAT	N/A	never raised
plugins. file	.getHomeDirectory()	.getHomeFolder()	Match naming convention
	.getRemoteList()	.getRemoteFolderContents()	Match naming convention
plugins. http	.createHttpClient()	.createNewHttpClient()	The HTTP plugin was refactored to support all types of HTTP Requests and support more finegrained control over the requests. Instead of creating an HttpClient and referencing it by name, the HttpClient is now an object with it's own set of methods, for example methods to create all the different types of HttpRequests. Each XxxRequest object has a .executeRequest() function that returns a Response object again with it's own methods.
	.deleteHttpClient()	N/A	No longer required
	.getHttpClientCookie()	.getCookie() on HttpClient object	
	.getHttpClientCookies()	.getCookies() on HttpClient object	
	.getLastPageCharset()	replaced by .getCharset() on Response object returned by the .executeRequest() method on any of the XxxRequest objects created by the .createXxxRequest() methods on the HttpClient object	
	.getPoster()	.createPostRequest() on HttpClient object	
	.put()	.createPutRequest() on HttpClient object	
	.setClientProxyUsernamePassword()	.setClientProxyCredentials() on HttpClient object	
	.setHttpClientCookie()	.setCookie() on HttpClient object	
plugins. window	setToolBarVisible()	.setToolBarAreaVisible()	Name now matches what the function does
MenuBar Toolbar	.validate()	N/A	No longer required

Utilizing the new Solution Design features

Servoy 6 introduces many new features for designing solutions. This chapter will only highlight the new features and provide links to extended documentation.

New features

Windowing API

Windows and Dialogs are now instances of JSWindow, that exposes it's own API

Drag 'n' Drop

The Drag 'n' Drop implementation was updated with a onDragEnd event handler that fires with either a drop occurred or the drag operation was canceled

The Drag 'n' Drop related event handlers now get passed an JSDNDEvent instead of a regular JSEvent. The JSDNDEvent exposed an extended API compared to the JSEvent API, exposing functions and properties that are relevant only in Drag 'n' Drop operations, for example the recordIndex on which the drop took place.

Inheritance

The inheritance model was improved in the UI inheritance area, to allow overriding the inherited properties on Form, Form Part and element level.

Encapsulation

On Forms an encapsulation property was added to make it possible to hide parts of the Forms scripting API. The following objects can be hidden:

- elements
- foundset
- controller

- all dataproviders

Additionally, it's possible to set the Form overall visibility:

- Private: the Form is only visible at design-time in Servoy Developer and can be added to container elements like TabPanels or splitpanes. At run-time the Form is not accessible in scripting.
- Module Private: Like Private, but the Form is visible in scripting within it's containing Solution

On the scripting level, the JSDoc tags @private and @protected are introduced to allow encapsulation on functions and variables:

- @private: Variables and functions marked as private can only be seen within the scope (.js file) in which they are declared
- @protected: Variables and functions marked as protected can only be seen within the scope (.js file) in which they are declared AND all child scopes: This is applicable in Servoy when using Inheritance on Forms: protected variables and functions declared on the super form can be accessed from the child forms

CSS Row Styling

Servoy 6 supports styling the odd, even and selected rows in grids (Portals and the body of Forms in TableView) using CSS:

```
odd, even, selected {
  background-color: #FFF;
  color: #5C5C5D;
}
odd {
  background-color: #F5F7F9;
}
even {}
selected {
  background-color: #d4d4d4;
  color: #FFF;
}
```

This functionality is a replacement for the deprecated "rowBGColorCalculation" property on Forms when it comes to plain odd, even & selected row coloring.

onRender event

The new onRender event on Forms, Portals and individual elements allows dynamic conditional styling of the object that is rendered.

See [Conditional styling using the onRender event](#) in the [Styling Solutions](#) chapter of the [Programming Guide](#) for more information

SolutionModel extensions

The SolutionModel API has been extended with the following API categories:

- removeXxx functions to remove objects from the Solution design
- createXxx factory functions to generate the string values for borders, pageFormat and fonts
- Bean support: ability to work with Beans that are on forms or to add new Beans to Forms
- Calculation support: ability to get, modify, remove and add Calculations
- getUUID() method on all JSXxx classes to retrieve the internal identifier of the Object

DatabaseManager extensions

- .addTrackingInfo(): Ability to provide an additional value that gets inserted into the specified column in the log table when tracking is enabled
- .createDataSourceByQuery(): Shortcut to databaseManager.getDataSetByQuery(...).createDataSource(..). The new function has considerable less overhead as the two-step alternative of first creating a DataSet based on an SQL statement and then creating a DataSource out of it
- .dataSourceExists(): method to check if a certain datasource exists

Tablefilter changes

The behavior of databaseManager.addTableFilterParam() has been changed, see [Changes that might break existing code](#)

visible & enabled Designtime properties

All standard UI components in Servoy were extended with designtime visible & enabled properties.

The visible property can be used

.border, .font, .titleText, .imageUrl properties instead of only a getter or setter

RuntimeXxxx types

Each UI component, including beans are now exposed as Types in the Scripting layer. All standard UI component in Servoy have a type that is prefixed with "Runtime", for example RuntimeForm, RuntimeButton etc.

All standard UI components also extend RuntimeComponent

Extended scripting API for RuntimeGroup

The scripting API of a grouped set of elements has been extended. The RuntimeGroup also extends RuntimeComponent

Solution deeplink normalization

The mechanism of sending arguments into a Solution on startup has been normalized.

- The arguments are now send into both the onOpen event handler of the Login as well as the main Solution
 - The use of short ("a") or full syntax ("argument") when passing the argument values has been made transparent: short syntax will automatically get converted to full syntax
 - Non-arguments like the deeplink method name or solution name are filtered out
 - Proper support for specifying multiple values for both the predefined "argument" parameter as well as custom named parameters
- Examples:

- ```
http://domain:port/servoy-webclient/ss/s/{solutionName}/argument/1/m/{globalMethodName}/a/2/a/3/b/1/b/2/b/3 or
http://domain:port/servoy-webclient/ss?s={solutionName}&a=1&method={globalMethodName}
&a=2&a=3&b=1&b=2&b=3 or
http://domain:port/servoy-client/{solutionName}.jnlp?a=1&a=3&b=1&a=2&b=2&b=3
```

all results in the following arguments applied to the onOpen method

```
arguments[0]: 1
arguments[1]: {argument: [1,2,3], b: [1,2,3]}
```

## Extended ClientDesign

Through UIProperties the available ClientDesign operations on elements can be restricted. See [Client Design mode](#) for more info.

## Tooltip display settings

UI Properties have been added to control the show and dismiss delay for ToolTips in both the Smart and Web Client.

See [TOOLTIP\\_DISMISS\\_DELAY](#) & [TOOLTIP\\_INITIAL\\_DELAY](#)

## i18n additions

The i18n object has been extended with a .setTimeZone(), .getAvailableTimeZoneIDs() & .getCountries() method

## HTTP plugin updates


The HTTP plugin is completely overhauled, to support all types for HttpRequest and provide fine-grained control over the request.

## Restful plugin updates

- ws\_authenticate: The Restful plugin have been extending with support for an ws\_authenticate method, allowing developers to roll their own authentication mechanism for exposed services.
- query parameters: Additional query parameters in requests are send into the ws\_\* methods as a last parameter
- Control HTTP\_Status of the HttpResponse: throwing an exception with one of the HTTP\_STATUS codes inside the ws\_\* methods will set the status code of the response of the webservice to the thrown status code. The HTTP\_STATUS codes are available through the HTTP\_STATUS constants of the HTTP plugin

## File plugin updates

The File plugin is extended with streaming functionality to send files back and forth between a Smart Client and the Servoy Application Server

 The changes to the File plugin already appeared in maintenance versions of Servoy 5.2, but are mentioned here for completeness.

## Maintenance plugin updates

The JSClientInformation object that is part of the Miantenance plugin was extended with a .getLoginTime() and .getIdleTime() method

## OpenId plugin

A new OpenID plugin is added to the standard plugin extensions of Servoy. The OpenID plugin allows to implement OpenID authentication in the Web Client