

New in 7.4

Mobile Client

- [SVY-4448](#) Added support for related find/search
- [SVY-4989](#) Enhanced the debug mode of the Mobile Client to read the Solution design live from Servoy Developer
Since Servoy 7.3 code changes made while debugging are synced between the debug client and Servoy Developer. However, if the debug session would end and then restarted, the debug client would not have the latest code. In Servoy 7.4 the debug client reads the latest code from developer when launched, so it will always have the latest code, instead of having to do a full export if there was a code change made since the last export
- [SVY-5336](#), [SVY-5164](#) Improved Mobile Export Wizard
The Mobile Solution Export Wizard has been improved by remembering all settings of the previous export and allowing to press the Finish button on the first page of the Wizard if there was a previously successful export that can be repeated based on the stored settings.
- Switched to JQuery Mobile virtual click events to work around the standard 300ms delay for click events on mobile browsers
- Improvements to ListView performance
- [SVY-5406](#) Added option to configure the order of inclusion for additional JS & CSS resources
See the Mobile Solution Export Wizard
- [SVY-5528](#) Added option to not auto-sync on first log in
If the first form of the Mobile solution is not bound to a datasource, then the Mobile Client will not automatically perform a sync if the Solution is started for the first time. It is then up to the developer to call the sync method before accessing data. This way, the developer has control over how and when to perform synchronization and error handling in case of issues during the sync.

Servoy Developer

- [SVY-3659](#) HTML-based Form Editor (currently only in use for Servoy Mobile)
By default Servoy Mobile Forms will still open in the existing Form Editor. To enable the HTML-based form Editor, go to Window > Preferences > Servoy > Form Editor and uncheck the "Use classic form editor for mobile forms" checkbox.
Note that by default the browser provided by the operating system is used as internal browser in Servoy Developer. For the best experience it is advised to replace the default browser with the Mozilla XulRunner browser component. See [Install Mozilla XulRunner as internal browser](#) for the steps involved.
- Made Mozilla XulRunner available as plugin to replace the OS-specific browser using as internal browser inside Servoy Developer
See [Install Mozilla XulRunner as internal browser](#) for the installation steps.
- [SVY-3518](#) Upgraded to Eclipse 4.3.1
 - Color preview on hover over color declarations in the CSS Editor
 - [SVY-5252](#) Support for CSS3 in the CSS Editor: this does not enable CSS3 usage in StyleSheets used on Forms, but allows setting a CSS3 Profile on .css files stored in the media library.
To set the CSS Profile, open the Navigator View, locate the .css file, select 'Properties' from the context menu > Web Content Settings > select desired CSS Profile)
- [SVY-170](#) Added support to group Forms together in the Solution Explorer
The context the `Forms` node of the Solution Explorer provides an entry to "Add working set". Through this option a 'folder' can be created, into which Forms can be dragged and dropped.
The folders or working sets are stored in the Resources project and thus can be shared with other developers through the team provider used
- [SVY-5033](#) Added ability to drag and drop media entries around folders inside the media library
- [SVY-5202](#) Grouped layout related properties in the Properties View
- [SVY-4448](#), [SVY-5195](#) Improved deleting of User Groups by allowing multi-select
- [SVY-375](#) Improved the "Select Dataprovider" dialog with option to create new scope variables
- [SVY-5145](#) Exposed encapsulation and deprecation properties in the ValueList and Relation Editors
- [SVY-3149](#) Ability to locate scopes using the Servoy Locator
- [SVY-132](#) Implemented Search for References on entries in the Media Library
- [SVY-5257](#) Better indication of what type of update is available through the auto-update mechanism
The Servoy Developer entry in the overview of available updates will have a version that includes details on whether the update is a release candidate or final and similar information is provided in the Description area of the overview.
Additionally, the non-final updates are exposed through a new update site, which is disabled by default and should be enabled to receive updates for release candidate updates (see Window > Preferences > Install/Update > Available software Sites). The new update site will be available in new Developer installations from Servoy 7.4 onwards. In Developer installations that were created with Servoy versions prior to Servoy 7.4, the update site can be added manually. The url is https://www.servoy.com/developer/70x_updates/releasecandidate
- [SVY-5194](#) Improved error reporting when attempting to work on Solutions that are created/modified using a newer version of Servoy Developer

Testing & Debugging

- [SVY-5210](#) Improved display of Excepted vs. Actual in case of failing UnitTests run inside Servoy Developer
- [SVY-3630](#) Support exporting to .servoy files from both the Export wizard in Servoy Developer as well as the command-line exporter, based on the .dbi files in the workspace, ignoring the actual database structure
This feature is especially helpful when doing automated exports through the command-line, for example in a software factory setup
- [SVY-5195](#) Fixed initialization of Solution when relaunching Debug Smart Client
Prior to this fix, in certain scenarios not all variables in scopes were initialized again in the newly launched Debug Smart Client
- [SVY-5613](#) Support for debugging self-executing functions (IIFE) assigned to variables
Self-executing functions (or IIFE's) are functions that immediately execute themselves. This can be used to do initialization when a scope loads for example. Previously breakpoints inside such functions would never get hit.

```
var init = (function(){
    //your code here
})();
```

Note that the outer parenthesis are not needed, but are considered a proper code convention for IIFE's

Build system/JSDoc enhancements

- SVY-5532, SVY-5523, SVY-5531, SVY-5527 Improved support for JavaScript prototype inside code
Servoy's Script Editor and Build system now have good support for JavaScript prototyping. This allows creating JavaScript objects using prototyping and having proper code completion and builder markers. Supports both setting an Object as prototype or directly assigning new members to the prototype. Prototype members can be marked as deprecated or protected through JSDoc annotations.

```
/**
 * @param {String} name
 */
function BaseEntity(name) {
    /**
     * Storing name as a protected instance variable
     * @protected
     */
    this.name = name
}
/**
 * Self executing function (IIFE) to setup the prototype of BaseEntity when the scope in which the
 * functions reside gets instantiated
 *
 * @private
 * @SuppressWarnings(unused)
 */
var initBaseEntity = (function() {
    //Setting the prototype of BaseEntity to an object with a set of methods
    BaseEntity.prototype = {
        publicMethod: function() {},
        /**
         * @protected
         */
        protectedMethod: function() {},
        /**
         * @deprecated
         */
        deprecatedMethod: function() {}
    }
    //Extending the previously set prototype object with another method
    BaseEntity.prototype.getName = function() {
        return this.name
    }
})();

/**
 * The extends tag signals the build system that public & protected members added through super
 * constructor are known to code completion and the build system<br>
 * <br>
 * The constructor tag takes care of removing the inconsistent return value warning<br>
 * <br>
 *
 * @constructor
 * @extends {BaseEntity}
 * @param {String} name
 * @param {String} type
 */
function ExtendedEntity(name, type) {
    //Fail-save for when the ExtendedEntity gets called without the 'new' keyword
    if (!(this instanceof ExtendedEntity)) {
        return new ExtendedEntity(name, type)
    }
    //Calling the BaseEntity constructor, so that the logic defined in the constructor is invoked
    BaseEntity.call(this, name)
    /**@protected*/
    this.type = type
}
```

```

/**
 * Self executing function (IIFE) to setup the prototype of ExtendedEntity when the scope in which the
 functions reside gets instantiated
 *
 * @private
 * @SuppressWarnings(unused)
 */
var initExtendedEntity = (function() {
    /* Setting the prototype of ExtendedEntity to an object that has BaseEntity.prototype
 as prototype
    * BaseEntity.prototype is not used directly as prototype for ExtendedEntity, as this
 would mean that any additions made to
    * the prototype of ExtendedEntity would actually be made on the prototype of
 BaseEntity
    */
    ExtendedEntity.prototype = Object.create(BaseEntity.prototype, {})
    //Properly set the constructor
    ExtendedEntity.prototype.constructor = ExtendedEntity
    ExtendedEntity.prototype.getType = function() {
        return this.type
    }
})();

function test() {
    var x = new ExtendedEntity('Servoy', 'company')
    application.output(x.getName()) //Yields 'Servoy'
    application.output(x.getType()) //Yields 'company'

    //These bits of code will result in warnings
    x.protectedMethod()
    x.deprecatedMethod()
    x.name
    x.type
}

```

- [SVY-5615](#) Improved build system to handle special JavaScript methods like `function.call`, `function.apply`, `function.bind` and `Object.create`. For `.apply/call/bind`, the build system will recognize that the `.apply/call/bind` method will return the same type as the function on which it is called, for example:

```

var x = Object.prototype.toString.call(object) //Build system will know that .call will return a
String, as it is called on the .toString() method of Object, which returns a String value

function test() {
    var y = Array.prototype.slice.call(arguments) //Build system will know that y is an Array, as .
slice() of Array returns an Array
}

```

For `Object.create(object, properties)` the build system will know that what `Object.create` returns has the same type as the value of the object parameter, enhanced with the (optional) properties (See [Object.create](#) for more info)

- [SVY-5827](#) support function types with rest parameters in typedefs

```

/**
 * @typedef {{
 *     name: String,
 *     handler: function(String, Number|*...)
 * }}
 */
var MyType

```

- [SVY-5114](#) Improved support for Union Types in JSDoc. For example function parameters can now be declared to take an Array containing Strings and/or Numbers.

```

/**
 * Method defined to take one argument, which is an Array that can contains both Numbers and Strings
 * @param {Array<String|Number>} arg
 */
function method(arg) {}
function demo() {
  //These are fine
  method([1, 'one', 2, 'two', 3, 'three'])
  method([1,2,3])
  method(['one','two','three'])

  //This invocation raises a builder marker, as false is neither a Number or a String
  method([1, 'two', false])
}

```

- [SVY-5113](#) Support builder markers when supplying a reference to a function object as value to another functions parameter, but the signature does not match

```

/**
 * @param {function(String, Number):Boolean} f
 */
function method(f) {
}

function demo() {
  /**
   * @param {String} name
   * @param {Number} age
   * @return {Boolean}
   */
  function one(name, age){
    return true
  }

  //No warnings here because the signature of function 'one' matches the requires signature for
  the 'f' parameter of 'method'
  method(one)

  /**
   * @param {Number} age
   * @param {String} name
   * @return {Boolean}
   */
  function two(age, name){
    return true
  }

  //These raise builder markers for obvious reasons
  method(two)
  method(function(){})
}

```

- [SVY-3555](#) Enabled the strike-through of deprecated member declarations in the Script Editor
- [SVY-5524](#) Exposed Error.stack property in scripting
Allows getting the stacktrace of JavaScript Error objects
- [SVY-5371](#) Support returning an instance of itself inside Constructor functions without warnings being generated
This allows building in a fail-save for Constructor function not being called with the new keyword

```

/**
 * @public
 * @constructor
 */
function MyConstructor(name) {
    if (!(this instanceof MyConstructor)) { //constructor is not called with the 'new' keyword
        return new MyConstructor(name)
    }
    //rest of the constructor logic
};

```

- [SVY-3049](#) Made the preference to initially fold the Header comment work (Preferences > JavaScript > Editor > Folding)
- [SVY-4876](#) Fixed the generation of JSDocs through the "Generate Element Comment" option to not insert an @return tag if the function does not return anything
- [SVY-4226](#) Improved "move code" option of the Solution Explorer to not insert the scope prefix if inserting into the same scope

Solution Development



Behavior Changes

[SVY-5618](#) In Servoy 7.4 the behavior of passing custom exceptions thrown from JavaScript into the Solutions onError handler has been changed due to a bugfix.

Prior to Servoy 7.4, the custom exception object was passed into the onError handler wrapped in a undocumented Java class. As of Servoy 7.4 the actual thrown object is passed directly into the onError handler. Implementations that have worked around the bug.....

[SVY-5538](#) Behavior Change in the Web Client due to aligning the behavior of controller.enabled with the Smart Client behavior

After disabling a controller, individual elements on the controller can now be enabled through scripting. This behavior has been present in the Smart Client for a long time. The behavior in the Web Client has now been brought inline.

[SVY-5213](#) Prior to Servoy 7.4 the method application.getValueListDisplayValue would not return a result for real values that are not in the first 500 entries in large ValueLists. As of Servoy 7.4 the display value is always returned, regardless on the size of the ValueList

- [SVY-2648](#) Added ability to set imageUrl in onRender
- [SVY-5443](#) Added support to the rawSQL plugin to get all ResultSets returned by executed Stored Procedures
- [SVY-4134](#) Support transparent dialogs
A JSWindow now supports a boolean transparent flag. When set to true prior to showing the JSWindow, the JSWindow itself will be transparent, so if it contains a (semi-)transparent form, the Ui underlying the JSWindow will shine through. Additionally JSWindow supports a opacity setting, which can be set with a value between zero and one, with zero meaning fully transparent and 1 meaning fully opaque. When set to a value less than 1, the entire JSWindow, including it's chrome and all the elements on the form will become semi-transparent
- [SVY-5660](#) JSFoundSet iterator support for easy looping through a JSFoundSet, making sure all records are processed that were present when starting the iterator (except when deleted during the iteration)
See `JSFoundSet.forEach(function)`
- [SVY-5837](#) Allow SQL statements that begin with 'with' instead of 'select'
Supported on:
 - `foundset.loadRecords(sql)`
 - `databaseManager.getDataSetByQuery(sql)`
 - `databaseManager.addTableFilterParam(datasource, column, 'in', sql)`
- [SVY-4685](#) Support for getting typed foundsets without having to resort to JSDoc typing
See `datasources.db.udm.contacts.getFoundSet()` for example

Web Client



Behavior Change

The behavior of controller.enabled in the Web Client has been brought inline with the behavior of the Smart Client. It now allows elements to be individually enabled even if the controller is disabled.

- [SVY-69](#) Disabled text-selection while a Drag 'n' Drop operation is taking place
The side effect is that selecting text on elements that also have an onDragStart event handling that starts a Drag 'n' Drop event is not possible anymore
- [SVY-521](#) Added ability to customize internal icons of Calendar and Image fields
See [Replacing Default Element Images](#)

- [SVY-1419](#) Made Calendar popup style able
See [Replacing Default Element Images](#)
- [SVY-5176](#) Improved onRender performance on consecutive calls
onRender will now only cause client side updates if there are actual changes to make.
- [SVY-5701](#) Updated HTML Editor (in editable HTMLArea) to TinyMCE
The old editor was replaced as it was not supported anymore and caused issues on newer browsers
The TinyMCE integration is a basic version of TinyMCE. It can be customized through the `APP_UI_PROPERTY.HTML_EDITOR_CONFIGURATION` client property of the HTML Area element
- [SVY-5774](#) Option to hide the loading indicator separately from the blockInputOnRequest
Whether or not the Loading indicator is visible can now be controlled through the `servoy.webclient.hideloadindicator` setting on the Servoy Admin page. Prior to Servoy 7.4 when enabling the `servoy.webclient.blockinputonrequest` setting, the Loading indicator would be disabled automatically.
- [SVY-5730](#) Added Arrow Up/down keyboard navigation to TableViews

Smart Client

- [SVY-4946](#) Improved UX of Smart client launch
Servoy 7.4 introduced 2 new server side settings that can be set to control the launch experience of the Smart Client:
 - `servoy.branding.loadingbackground`: Sets the background-color of the main Smart Client window when no form is showing. This is without other settings before and after a custom log in form is shown or while the default log in dialog is shown.
 - `servoy.branding.hideframewhileloading`: hides the main Smart Client window while no form is showing. This is before and after a custom log in form is shown or while the default log in dialog is shown.
 These two new settings can be used in conjunction with other branding related settings and can be set on globally or in Profiles

Deployment



Behavior Change

[SVY-5695](#): Since Servoy 6.1 the name of the Solution was automatically appended to the name of the shortcut created by Java Webstart when branding was enabled, to get the same behavior as when branding was not enabled and to be able to have multiple shortcuts be created for multiple solutions hosted on the same Servoy Application Server. However, this change did not take into account the fact that if only one solution was hosted, it might be preferred to not have the name of the Solution included, but only the value of the `servoy.branding.webstart.shortcuttitle` setting. It also did not take into account the fact that the name of the Solution is usually something internal to the developer and the public "name" is set in the title property of the Solution. Hence Servoy 7.4 reverts the behavior of including the name of the solution into the title of the shortcut by default when branding is enabled. Instead it now supports to put the value of the title property of the Solution into the name of the icon, by including the following syntax in the value of the `servoy.branding.webstart.shortcuttitle` setting: `%%solution.title%%`

[SVY-5876](#) In the case of exceptions occurring in the Web Client, the default error page offers a link to return to the homepage. Prior to Servoy 7.4 this link would redirect the user to the main entry point of the Servoy Application Server. As of Servoy 7.4 this instead redirects to `/servoy-webclient/`

- [SVY-3072](#) Added support for exporting the scale of columns when exporting to a .servoy file
- [SVY-5357](#) Ability to include Solutions in a WAR export
See the additional checkbox on the War Export Wizard (Solution Explorer > Active solutions > Context menu > Export Solution > War Export)
- [SVY-5774](#) Introduced `servoy.webclient.hideloadindicator` setting to control the display of the Loading Indicator independent from the `servoy.webclient.blockinputonrequest`

Plugin & Bean Development

- [SVY-5242](#) Improved how Servoy scans jar files in the plugins & bean folders to improve performance
See [Creating client plugins - Entry Points](#)