SuperReport Pro™

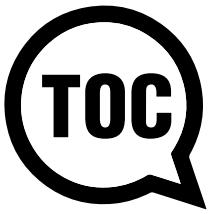
Developer Manual

Version 3.4

e-Node
30 rue de la République
33150 Cenon
France

www.e-node.net

Manual revision: 16 March 2018



Contents

About SuperReport Pro

What is SuperReport Pro, and what can I do with it?

SuperReport Pro is a plugin for 4D which provides an enhanced report creation and printing tool for developers and end-users.

Using the SuperReport Pro Editor and plug-in routines, developers can create sophisticated reports that support a variety of data formats, including 4D fields, variables, and arrays.

In addition, SuperReport Pro can be integrated into any 4D application's user interface, providing your users with the ability to create reports using a number of advanced features not available using the standard reporting tool built into 4D.

SuperReport Pro's greatest benefit is the fact that it provides the ability for users to create and modify report designs long after a 4D database application has been completed without requiring any code changes to the database.

This benefit is further enhanced in the case of compiled databases, where SuperReport Pro allows reports to be changed without requiring either database changes or recompilation.

SuperReport Pro reports can be held as documents on disk, or be contained within the data file belonging to the database. Thus vertical market applications sold to many customers can be supplied with standard reports, which can later be customized by the user.

Since these reports are held in the data file or external files, program updates installed at a later date will have no impact on the users' systems - the same program update can be sent to all users and each user's customized reports are preserved!

SuperReport Pro's power does not come at the expense of simplicity. Most users will be able to grasp the concepts involved in designing reports, since the editor is so similar to a drawing program.

Report operations like break processing, which can be cumbersome and complex in 4D, are also made simpler, with the user able to specify when breaks occur and which objects are to be totalled, without requiring any code.

Items that are new or modified in SuperReport Pro version 3.2 are displayed in pink (magenta) characters.

Items that are new or modified in SuperReport Pro version 3.3 are displayed in green characters.

SuperReport Pro Features

SuperReport Pro includes a wide variety of features for creating fully customizable reports. Using the full-featured plug-in area, you can quickly and easily enhance your application's reporting capabilities.

If the standard functionality is not enough, you can take advantage of the complete developer API to further extend the power of SuperReport Pro.

Included in SuperReport Pro™ are the following features:

- Compatibility with 4D v11 to v15, including 64-bit 4D Server.

Note: on Mac, only Intel processors are supported.

- Include a full-featured reporting tool in any application, quickly and easily
- Enhance reporting capabilities using the SuperReport Pro developer API
- Complete support for creating HTML formatted reports
- Repeating objects for printing related many records
- Supports printing 4D arrays, including direct access to specific array elements
- Support for executing callback routines and object scripts when printing reports
- Support for executing custom scripts for enhanced script management
- Support for custom structure views – great for isolating which tables and fields the user can access
- Developer API for customizing end-user functionality
- Enhanced Editor Customization
- Print SuperReport Pro areas, or save them as HTML

Technical Details

Compatibility Information

SuperReport Pro Version 3 is compatible with 4D v11 to v15, for both MacOS and Windows (including 32-bit and 64-bit servers). It requires MacOS 10.7.5 or higher and Windows 7 or better.

Technical Support

Technical support for SuperReport Pro is provided via the online web forums.

Installation

Installing the plugin

SuperReport Pro is provided as a bundle for both Windows and MacOS: there is just one version for both platforms. To install it, simply copy the file **SRP.bundle** into your Plugins folder.

Plugins folders can be located in one of two locations:

- In the 4D application folder (4D or 4D Server). When plugins are installed in this location, they will be available to every database that is opened with that application.
- Next to the database structure file for your project: in this case, the plugin will only be available to that database. On MacOS, this means that the Plugins folder must be placed within the database package or folder. To open a package, ctrl-click on the package and choose **Show Package Contents** from the contextual menu.

Using SuperReport Pro in Demo mode

You can use SuperReport Pro in Demo mode for 20 minutes, after which time it will cease to work. When this becomes annoying, it's time to buy a license, which you can do on our website.

Licenses are either linked to the 4D product number, the workstation or the company name as described below.

Licensing

Like all e-Node plug-ins, SuperReport Pro offers several license types. There are no such things as MacOS vs Windows or Development vs Deployment.

For current pricing, please see the ordering page on our website.

Definitions

Regular and merged

- **Regular licenses** are used for applications that are opened with 4D Standalone or 4D SQL Desktop, or with 4D Server, either in interpreted or compiled mode (doesn't make a difference regarding plugin licensing).

  These can be either single user or server databases and they are linked to the 4D or 4D Server license: you need to provide the number returned by the "Copy" or "eMail" buttons from the plugin demonstration mode alert (this number is actually the 4D command **GET SERIAL INFORMATION** first parameter). This number is a negative long integer such as -1234567.

- **Merged licenses** are used for double-clickable applications built with 4D Volume Desktop (single user) or with 4D Server by means of the 4D Compiler module.

  These licenses are linked to the machine ID (single user workstation or server): you need to provide the number returned by the "Copy" or "eMail" buttons from the plugin demonstration mode alert (this number is calculated from the single user or server machine UUID). On 4D Server any remote client will return the server number. This number is a positive long integer such as 1234567.

In both cases the demonstration mode dialog will display the proper number according to the current setup (regular or merged) and the "Copy" and "eMail" buttons will use it as well.

License keys

- **Final licenses keys** are delivered by e-Node once you have provided the associated number as described above (4D serial information or machine ID). They activate the plugin either though 4D code or the Register button from the demonstration mode dialog.
- **Master keys** are delivered upon order if you opt for the Online instant activation system. The final license key is self-generated by the plugin and stored into the license file, so you don't have to bother with 4D serial information or machine IDs.

Free updates

- **Regular licenses**. A new license will be supplied for free at any time (maximum once a year) if you change your 4D version or get a new 4D registration key for the same version, provided that your previous license match the current public version at exchange time. This rule applies whether you are already using the new version or not: just specify that you also want a key for the older version as well as the current one when you order an upgrade.
- **Merged licenses**. These licenses are independent from the 4D versions and product numbers. They will remain functional if you upgrade e.g. from 4D v14 to 4D v15 on the same machine (single user workstation or server).

  You'll only need to update a merged license if your machine or motherboard is replaced (a new license will be supplied for free in this case, provided that your previous license match the current public version at the exchange time), or if you install a paid upgrade of the plugin.

Note: if you are using several concurrent versions of 4D you will need one plugin license for each version.

License types

- **Single-user.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) of applications that are opened with 4D Standalone or 4D SQL Desktop or built with 4D Volume Desktop.
- **Server.** These licenses allow development (interpreted mode) or deployment (interpreted or compiled mode, including merged servers / remotes) on 4D Server with up to 10 users (“small server”), 11 to 20 users (“medium server”) or more (“large server”).
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) on any number of 4D Standalone (or single user merged applications built with 4D Volume Desktop) that run your 4D application(s).

  It is a yearly license, which expires after the date when it is to be renewed. Expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire**.

  A single license key will unlock all setups on all compatible 4D versions and all versions of the plugin. The license key is linked to the developer/company name.

  This license allows deployment (selling new application licenses, updates or subscriptions) while the license is valid. **No new deployment may occur after expiry without a specific license** (merged or regular). End-users running deployments sold during the license validity period remain authorized without time limit, provided that they are no longer charged for the application using the plug-in (including maintenance or upgrades).
- **OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode, including merged) on any number of 4D Servers (any number of users), 4D Standalone or single user/remote merged instances that run your 4D application(s).

  It is a yearly license, under the exact same terms as the Unlimited Single User license described above, except that it also covers server deployments.
- **Unlimited OEM.** This license is a global OEM license, covering any combination of the plug-ins published by e-Node, including AreaList Pro, PrintList Pro, SuperReport Pro, CalendarSet and Internet ToolKit in all configurations.
- **Partner license.** This license matches 4D’s annual Partner subscription and covers all the plug-ins published by e-Node, including AreaList Pro, PrintList Pro, SuperReport Pro, CalendarSet and Internet ToolKit.

  For each product, a single registration key allows development (interpreted mode) or deployment (interpreted or compiled mode, except merged) on all 4D Standalones and 4D Servers (2 users) regardless of 4D product numbers, OS and versions. No merged applications.

  This is a yearly license, expiring on February 1st (same date as 4D Partner licenses). Expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire**.

Note: you don’t have to be a 4D Partner subscriber to subscribe to the e-Node Partner license.

Registering your SuperReport Pro License

Once you have purchased your license, you will receive a registration key. This code must be registered each time the database is started.

There are three ways to register your license:

- using the Demo mode dialog "Register" button,
- though a text file,
- in your 4D code with a command.

Both Register button and 4D code registrations can be performed in one single step through the Online registration system.

Yearly licenses such as Unlimited single user, OEM and Partner licenses do not require any serial information or online instant activation. The only way to register these licenses is through the SR_Register command.

Quick and easy way – End-user online instant activation

1. Make sure that the machine where the plugin will be used is connected to the Internet (single user workstation or in server mode the first remote client that will connect to the server).
2. Launch your application. Displaying any layout that uses the plugin will trigger the demonstration mode dialog.
3. Enter the Master key that was delivered by e-Node.
4. The plugin will display an alert indicating that it is now registered.

Note: this method does not require your source code to be modified or recompiled.

Quick and easy way – Developer online instant activation

1. Put the following lines of code into your **On Startup** database method, with the Master key that you received and your email address:

```
C_LONGINT ($result)
$result:=SR_Register ("yourMasterKey";0;"youremail@something.xxx")  // 0 if successful
```

2. Make sure that the machine where the plugin will be used is connected to the Internet (single user workstation or in server mode the first remote client that will connect to the server).
3. Install your application.
4. Launch your application. Displaying any layout that uses the plugin will silently (no dialog) register it.
5. You will receive an email with the final key that was issued and the IP address of the user site.

If the site has no Internet connection or if you want to use the plugin license system to help protect your own software copy, you can manage the final key registration yourself using one of the following methods.

The Demonstration mode dialog

The demonstration mode dialog is used for both Online instant activation and manual registration, unless the plugin is registrered with a final key or master key through the 4D code.

When using manual registration, single user and server licenses require that you first send us the relevant information (serial or machine ID, see Definitions).

Note: sending the serial information or machine ID is not needed with the Online instant activation system.

This action is performed from the Demo mode dialog, which is displayed upon the first call to the plugin.

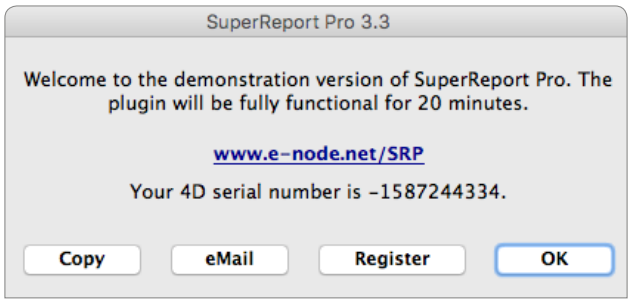
To trigger this display and enable your users to register without actually calling a command or setting up an area, you can also pass an **empty** string to SR_Register and the dialog will show:

```
C_LONGINT ($result)
$result:=SR_Register ("")  // display the dialog
```

Note: calling ***SR_Register*** with any key (valid or invalid) will not display the dialog.

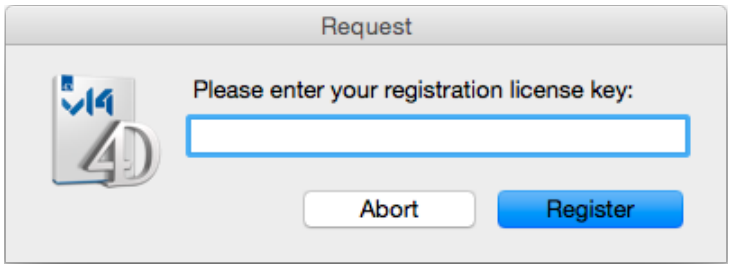
Retrieving the serial / machine information

The Demo mode dialog includes all relevant information (serial or machine ID, see Definitions) to obtain your license, as well as a “Copy” button to put this information into your clipboard or a text file, an “eMail” button to email the information to e-Node’s registration system and a “Register” button to enter your license key once received:

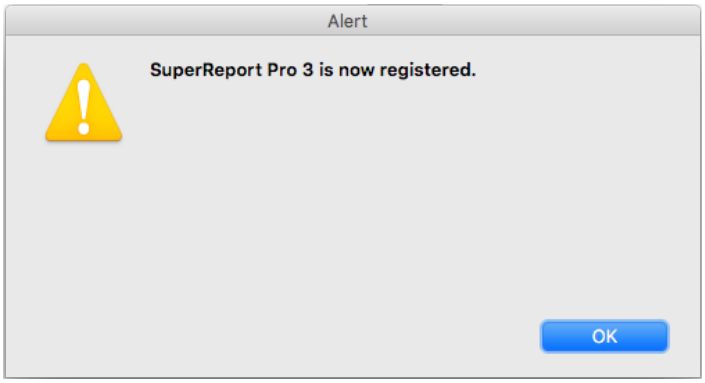


Using the “Register” button

Clicking on this button will display a standard 4D request to enter your registration key:



Paste or drag and drop your registration key and, if correct, the plug-in will be registered for all future uses on this workstation:



Note: if 4D does not activate the **Edit > Paste** menu item click **Abort** and **Register** again, or try drag and drop.

Note: you can directly paste the Master key that was delivered when using the Online instant activation.

Registering Server licenses

Similarly, server licenses can be registered from the demonstration mode dialog without having to modify your code and use SR_Register (which of course you can do with any license type). In this case, the 4D Licenses folder, serial information or machine ID used will only be the 4D Server information, not the client workstation's.

Server licenses can be registered on any client workstation (remote mode), or on 4D Server itself.

Registering in Remote mode

The server and all workstations can be registered from any single client workstation connected to the server. As in Single user mode, the Demo mode dialog will be displayed on a client workstation when one of the following conditions are met:

- Calling a SuperReport Pro command other than ***SR_Register*** with a non-empty parameter
- Calling ***SR_Register*** with an empty string

Use the **Copy**, **eMail** and **Register** buttons just as above and your server will be registered for all workstations.

Note: any other workstations previously connected (before registration occurred) will need to re-connect to the server to be functional.

Registering on 4D Server

To directly register the server and all workstations from the server machine itself, you need to display the Demo mode dialog on the server.

Call ***SR_Register*** with an empty string in the **On Server Startup** base method:

```
C_LONGINT ($result)
$result:=SR_Register ("")  // display the dialog
```

Use the **Copy**, **eMail** and **Register** buttons just as above and your server will be registered for all workstations.

Note: the dialog will automatically be dismissed on the server after one minute in order not to block client connections (the server is only available to client workstations once the On Server Startup method has completed).

Merged licenses notes

Both methods can be either used with regular or merged servers and client workstations.

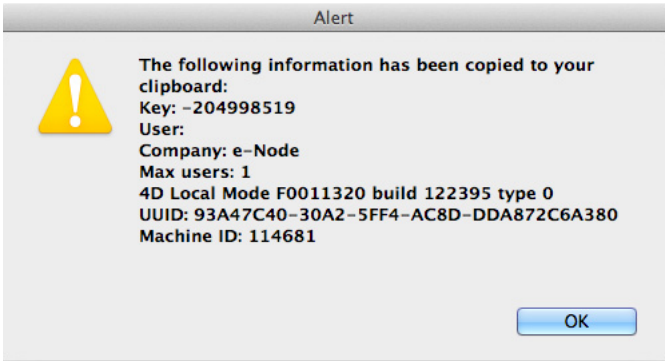
- Regular licenses are linked to the 4D Server serial information
- Merged licenses are linked to the 4D Server machine ID

Note: merged licenses will keep working if your 4D Server serial information is modified (upgrading or 4D Partner yearly updates), or if any client workstation hardware is changed.

It will only need to be updated if the 4D Server hardware is changed, or if the plugin itself requires a new key (paid upgrades upon major version changes).

You may want to register your merged server without having to turn off the database to modify the code. We have created a utility database to manage this - it's called Get Serial Info and you can download the appropriate version for your 4D version from the e-Node server.

This is possible using any 4D setup on the server machine (such as a standard developer single user 4D). Keeping your production server alive, open the Get Serial Info database with 4D on the same server machine. Ignore the demonstration mode dialog (if your single user 4D is not registered for the plugin) and wait for the next Alert:



A text file is also saved with the same information.

The last line "Machine ID" is the number that you need to send in order to receive your merged server registration key.

You can also check the machine ID in standalone mode (or on any remote client with the built-client application or in interpreted mode as long as it is running on the same server machine) with AreaList Pro using the following call:

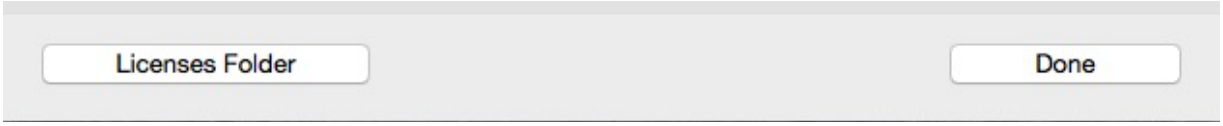
```
C_LONGINT($machineID)
$machineID:= AL_GetAreaLongProperty (0;"mach")
```

Note: you don't need an AreaList Pro license to do this.

Using a text file

Alternately, you can place a plain text file into your 4D Licenses folder.

To open this folder from 4D use the 4D Menu **Help > Update licenses,** then click the **Licenses Folder** button:



The text file **must** be called “SRP3.license4Dplugin” and be a plain text type file.

Just paste all your licenses for SuperReport Pro v3.x, one per line, e.g.:

MyLicense1
MyLicense2
MyLicense3

Any license type can be included into this document, except unlimited single user, OEM and Partner licenses.

Note: the Demo mode dialog **Register** button actually does this: create the text file and include the license key, or add the license key to the existing document if any.

Note: when using the Online instant activation system, the Master key is automatically converted to a Final key according to the current environment and this final key is stored into the license file.

Using SR_Register

1. Open the **On Startup** database method

2. Call the SR_Register function with your registration key - for example:

```
$result:=SR_Register ("YourRegistrationKey")  // result = 0 means registration was successful
```

If you have several licenses for different 4D setups you can call ***SR_Register*** multiple times in a row without further testing. See the Example with multiple calls.

Combining methods

When such a file exists in the Licenses folder SuperReport Pro will check for valid licenses from this document as a first action before anything else (including checking any ***SR_Register*** command).

If a valid license is included into the “SRP3.license4Dplugin” document any calls to ***SR_Register*** will return zero (for “OK”).

Therefore you can mix modes and use the text file (or **Register** button) as well as the command.

Unlimited single user, OEM, temporary and Partner licenses can only be entered through the ***SR_Register*** command.

Online registration

As of version 5.3, SuperReport Pro provides an automated solution to register itself using an Internet connection.

This feature can be helpful whenever you don't want to bother your end user with plugin registration, or want to save the time to collect the serial / machine ID, or any other reason when you expect the process to be entirely and automatically managed from the client site.

It can also be used for your own development tools, removing the need to modify your 4D code to include or update registration licenses.

Note: the site must have an open outgoing HTTP Internet connection available.

"Master" keys

The basic principle is that we deliver a non-assigned license key, called master key, which you use in your call to SR_Register in your **On Startup** database method. This key will be used to generate valid keys for the plugin and environment, called final keys.

One single master key can generate as many final keys as you need, in case you order several licenses of the same kind (regular or merged, single user licenses or server licenses of the same size).

A master key looks like a final key, except that the second part is the plugin code name (same as the license file name) instead of the serial / machine ID, e.g. "123456-SRP3-xyz".

Passing a master key as the first parameter to ***SR_Register*** when the plugin has not been previously registered by any of the methods above will result in a connection attempt to e-Node's license server as described below.

Master keys can also be entered by the user through the registration dialog. See Quick and easy way – End-user online instant activation.

Process

If the plugin has not been previously registered (through online registration, text file, register button or SR_Register with a final key), and if ***SR_Register*** receives a master key in its first parameter, it will recognize it a such, then:

1. Connect to e-Node's license server.

2. Ask the server if the master key has not been assigned yet (or if the master key is designed to generate several final keys, if there is any unassigned key up to that number).

3. Send the serial information (regular licenses) or the machine ID (merged licenses) to the license server.

4. If an error is detected (such as master key not matching the current setup) return an error to ***SR_Register***.

5. If the master key is valid, receive its final key from the license server then register itself (writing into the license file).

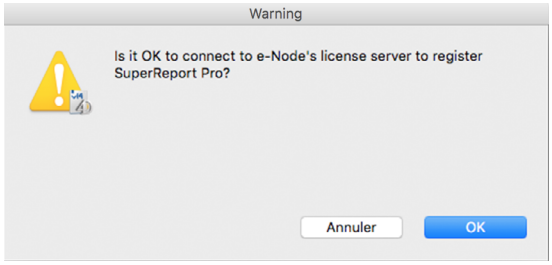
Note: if a final key has already been issued for this serial / machine ID using this master key, it is simply resent.

User interface

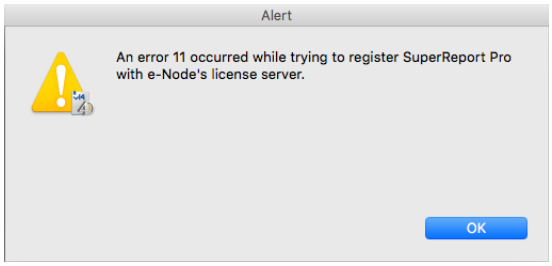
In addition, SR_Register second parameter allows optional settings regarding the user interface in the online registration process.

```
C_LONGINT ($result)
$result:=SR_Register ("Master key";0 ?+1 ?+2 ?+3;"youremail@something.com")  // all dialogs
```

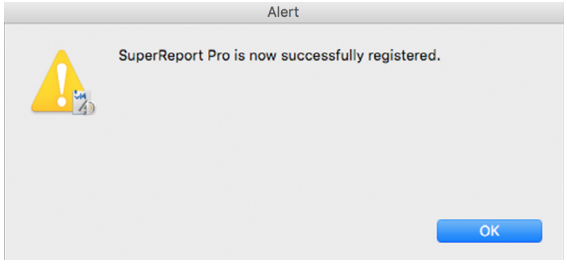
Display a confirmation dialog before step 1



Display an alert at step 4



Display an alert at step 5



eMail notification

The third parameter to SR_Register (optional) is the developer email to whom the information will be sent (if this parameter is used and non empty, of course).

The emailed information includes both the final key issued and the IP address from where it was requested (and to where it was sent for registration).

- When a key is issued:

  Title: SRP3 license

  Body:

  License 123456-123456789-abcdefgh

  issued to 12.34.56.78

- When a key is resent:

  Title: SRP3 license

  Body:

  License 123456-123456789-abcdefgh

  resent to 12.34.56.78

The default mode (master key being passed as the only parameter) is silent: no confirmation, no alert, no email.

Getting Started with SuperReport Pro

This chapter outlines the fundamentals of using SuperReport Pro, including:

- An overview of creating SuperReport Pro areas
- Customizing the SuperReport Pro area
- Extending SuperReport Pro with object scripts
- Understanding the SuperReport Pro Event Cycle
- Dealing with multi-platform issues.
- HTML support

For detailed information on using the SuperReport Pro commands, please refer to the Command Reference section.

See the Tutorial section in the User Guide to learn more about getting started with SuperReport Pro.

Creating SuperReport Pro Areas

There are two ways in which a SuperReport Pro area can be presented to your users:

1. Using the 4D **Open external window** command

2. Displaying a 4D form (either via a dialog or input form)

In addition, you can create and use offscreen areas, for programmed actions such as printing or exporting without user interaction.

Using an External Window

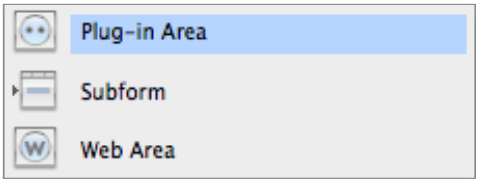
If you wish to use SuperReport Pro in an external window, you simply need to call the 4D **Open external window** routine:

```
C_LONGINT($areaRepRef)
$areaRepRef:=Open external window(50;50;400;400;8;"SuperReport Pro";"%SuperReport")
```

Using a 4D Form

The most common way of using SuperReport Pro is to place a plug-in object on your input form or dialog. To do this:

1. Select the Plugin Area option from the tool palette on your form:

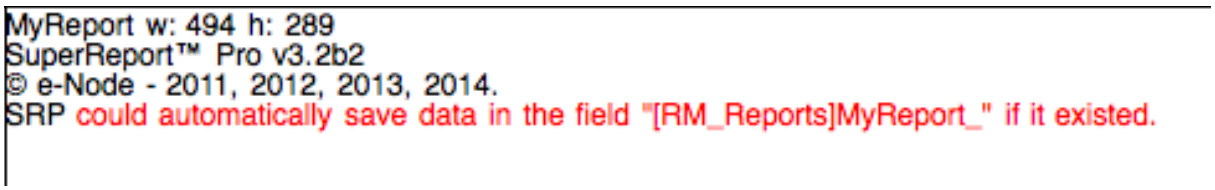


2. Draw a rectangle on the form in the size that you want your report to appear.

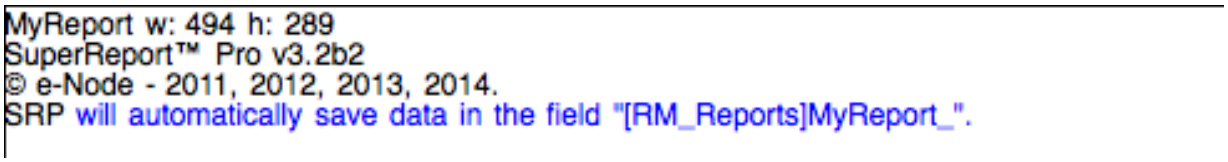
3. In the Properties palette, choose **SuperReport Pro** from the **Type** dropdown menu

4. Assign a name to the object variable

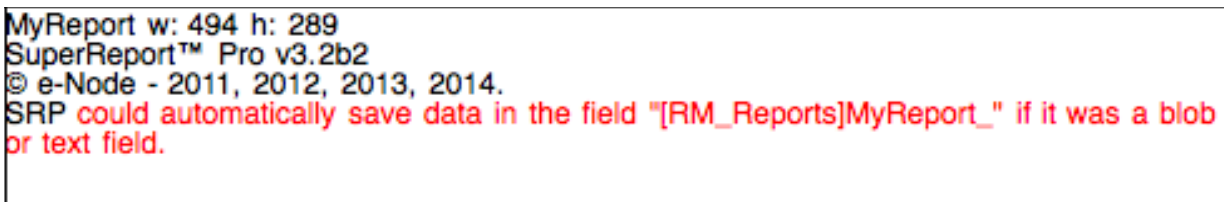
Your SuperReport Pro area will look like this:



Note that if you have a field in the table that has the same name as the report variable followed by an underscore, the comment changes:



The field must be a Blob or Text field. Otherwise, you'll see the following comment:



For more detailed information on using SuperReport Pro on an input form, please refer to the Tutorial section in the SuperReport Pro User Guide.

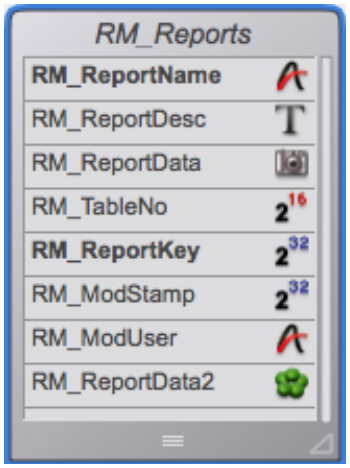
You can use the commands and functions to configure every aspect of an SuperReport Pro area, and to get information about an area. The commands and functions are grouped into themes such as Getters, Setters, Printing, etc.

Basic Integration

SuperReport Pro can be integrated into your applications via two different methods.

The first method uses a 4D external window; the second uses a 4D form that can be displayed in a dialog or input form.

All examples discussed in this section assume you have a custom table for storing the various reports you create. In our examples, we'll use the following table structure, which is defined in the Demo Database that is supplied with SuperReport Pro:



External Window

Using the 4D **Open external window** command, you can display the SuperReport Pro editor in a fashion similar to the **New process** command.

1. Open the SuperReport Pro editor in an external window using the following code.

```
C_LONGINT($areaRepRef)
$areaRepRef:=Open external window(50;50;Screen width-50;Screen height-50;8;"New Report";"%SuperReport")
```

When you display SuperReport Pro in an external window, you should use a Window type of 8 so that the window can be resized by the user and the close box is available. If you use a modal window (type 1 or type 5), the user will not be able to close the window.

2. If you wish to communicate with the SuperReport Pro editor in an external window, you can use the window reference ($areaRepRef) as the Area / Report reference parameter for those routines which interact with the plug-in area.

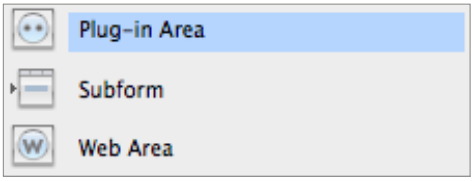
```
C_TEXT(reportPath)
reportPath:="Hard Disk:myReport.srp"
$error:=SR_NewReport ($areaRepRef;reportPath;1)
```

Plug-in Area

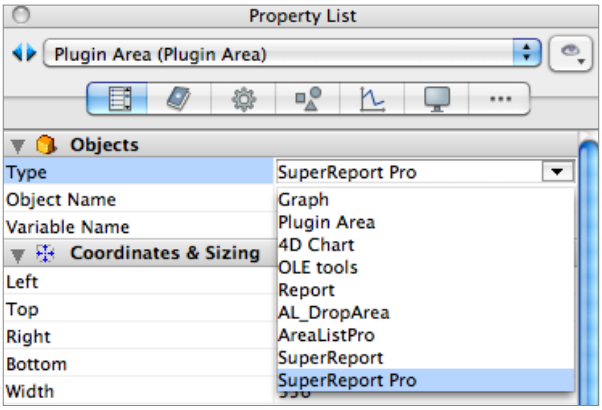
The second (and most common) way to use SuperReport Pro is to place the plug-in area on an input form, then provide access to the input form either via the **DIALOG** command or through the standard **MODIFY SELECTION** command (or User Mode access).

1. Create a new input form, or modify an existing input form, in which you wish to display the SuperReport Pro plug-in area.

2. Create a new plug-in object using the 4D Object List:



3. Select the SuperReport Pro object from the Plug-in Area popup menu to create a SuperReport Pro area.



4. Assign a name to your variable, which will be used by the related SuperReport Pro command to reference your report area.

If you define the name of the report area which is the same name followed by an underscore of a field which is in the table which contains the form you have created, SuperReport Pro will automatically load and save the report when the record is loaded.

For example, if you have a field entitled [Reports]ReportData_ when creating the SuperReport Pro area, using a name of "ReportData" will instruct SuperReport Pro to automatically load/save the report when the record is loaded and the form is displayed.

The type of this field should be a blob field.

If you are using the auto save feature of SuperReport Pro, you may proceed to Step 7.

If you are not using a "report" table, you will need to procedurally load/save the reports using the appropriate SuperReport Pro routines (Step **5** and Step **6**).

5. The next step is to create the report data so that your users can access it. The first **If** test determines if it's a new record. If so, we create a new report using the SR_LoadReport command. If it's an existing record, we load the data from the saved report record with ***SR_LoadReport*** (we'll cover how to save report data in a bit).

```
C_LONGINT($error)
Case of
   :(Form event=On Load)
      If (Is new record([Customer]))  //it's a new record
         $error:=SR_LoadReport ($areaRepRef;SR_GetTextProperty (0;0;SRP_Area_NewReport))
      Else
         $error:=SR_LoadReport ($areaRepRef; BLOB to text ([RM_Reports]RM_ReportData; \
            UTF8 Text without length))
      End if
End case
```

At this point, we are ready to begin using the SuperReport Pro editor. However, before we proceed, let's add the code necessary for saving your report information to our report data field so that we can access it again.

6. The last step we need to complete is saving our reports to the SuperReport Pro data field. The easiest way to do this is to make sure you have a field, appropriately named, to auto-save the report (see the discussion on this topic).

When you are retrieving information from a SuperReport Pro plug-in area, you must use a 4D text variable; passing a text field will not work correctly. Once the SR_SaveReport routine has completed successfully, you can copy the variable to your storage field as we have above.

7. Now you are ready to begin using the SuperReport Pro editor. Enter the User Mode (or however you have configured access to the [Report_Mgr] table) and create a new record. You should see something similar to the following input form:

Offscreen areas

An offscreen area is a plugin area that is only available in memory, not visible to the user.

Offscreen areas can be used for printing or exporting without any user interaction.

Such reports are created with SR_NewReport or SR_NewReportBLOB.

Make sure to always delete offscreen reports in order to free memory after you've finished with them, using SR_DeleteReport.

See for example Creating Reports Procedurally.

Upgrading from Previous Versions

SuperReport Pro version 3 is compatible with 4D versions 11 to 15.

To upgrade to SuperReport Pro version 3.x, simply install it as described in the Installation section of this manual, replacing your older version.

Major differences from previous versions

Compatibility Mode

You do not have to update all your SuperReport Pro areas and code immediately; SuperReport Pro version 3 will automatically run in compatibility mode.

Compatibility Mode Behaviour

When running in compatibility mode, the following behaviours are different:

1. ***SR RELATIONS*** sets compatibility mode on (in SuperReport Pro 3 it is a persistent property of a report, not a global one). In compatibility mode SRP_DataSource_RelateOne and SRP_DataSource_RelateMany are ignored in the report; the global settings are used
2. ***SR Set Script Callback*** when used with area = 0 sets compatibility mode on (in SuperReport Pro 3 it is a persistent property of the report, not a global one). In compatibility mode, SRP_DataSource_Callback in the report is ignored; the global setting is used.

What's Changed

Native Look

Now the appearance is always native.

New API

SuperReport Pro version 3 introduces a completely new API which is based on a full list of properties that the developer can get/set.

There are now fewer commands that you use to set and get an area's properties.

Each command affects just one property for the area, making your code much easier to understand and debug.

The new commands are organised into themes which relate to a particular part of the SuperReport Pro area, and some miscellaneous Utility commands.

For each theme there is a group of "Getter" functions and "Setter" commands, each targeting a different property type. For example, you can use the SR_GetRealProperty command to get the value of a property that has a real number value as its result.

Note that Boolean properties are called as longints (1 = True, 0=False). Don't worry though - you will not need to re-write all your SuperReport code.

Most of your existing commands will still work; the old commands act as wrappers for the new ones.

In fact you will still be able to write new code using the old commands, but if you want to take advantage of the new features, you'll need to use the new commands.

Some of the old commands are now obsolete or are no longer relevant and should be removed from your code.These are listed in the table below, along with details about how they should be replaced, where appropriate.

You can find a description of the new syntax in the Anatomy of a SuperReport Pro Command topic.

Saving Reports - now in XML

Looking at historical versions of SuperReport Pro, the earliest version saved reports as picture files. This was subsequently replaced with blobs, and now, in Version 3, they are saved as XML.

- You can use the SR_ConvertReportToXML command to convert existing reports from the old blob format into the new XML one. ***SR_ConvertReportToXML*** is used internally when needed - you don't have to convert all old reports to use them.
- SR_SaveReport creates an XML file or fills a variable with XML.
- SR_NewReport expects a text: the report as XML text or a path to the report in a file, which can be in old 2.x blob format.
- SR_NewReportBLOB expects a blob, which can be in old binary format.

Instead of **BLOB to text**, always use ***SR_ConvertReportToXML*** if there is any chance the blob contains old binary report.

If the report is already converted, it is immediately returned without conversion.

Or always use the blob API.

Using reports stored on disk is similar: **DOCUMENT TO BLOB** will work, but **BLOB to text** will fail if it is in old 2.x blob format.

But using SuperReport Pro's both old and new API to load a file will work.

Note that all the API using report as text is unusable in non-unicode mode due to the TEXT limit (32000 characters).

In such case, use the old API for loading / saving the report (or the new API when using files) and either old API or new blob API for printing / export.

To convert even older picture format documents first use the old command ***SR Report To BLOB*** to convert it from picture to BLOB:

```
$reportBlob:=SR Report To BLOB([Reports]PictureReport)
```

Updating Commands

The new API supports many more options and object types. You may want to replace calls to old functions with their new versions. Following are some examples:

Old Command	How to replace it

SR Get Scripts	Use the SR_GetPtrProperty command with the appropriate property:
	`$error:=SR_GetPtrProperty ($areaRepRef;$objNum;SRP_Object_Script;`
	`->objScript)`
	(See the Properties by Theme section for a list of all properties.)

SR Get Object Properties	Use ***SR_GetPtrProperty*** with the appropriate property - for example, to get the position of the top of an object:
	`objTop:=0`
	`$error:=SR_GetPtrProperty ($areaRepRef;$objNum;SRP_Object_PosTop;`
	`->objTop)`
	You can find a list of all the object property options in the Object Common Properties section.

SR Get Sections	Use ***SR_GetPtrProperty*** with the appropriate property. For example, to get the name of a section:
	`sectionName:=""`
	`$error:=SR_GetPtrProperty ($areaRepRef;$objNum;SRP_Object_Name;`
	`->sectionName)`
	You can find a list of all the section property options in the Section Properties section.

Obsolete Commands

If you are using any of the following commands, you will need to remove them from your code or modify them as explained in the Comments:

Old Command	Comments
SR Print PICT	Use SR_PrintIntoPICT or SR_PrintBLOBIntoPICT.
SR File Types	Supports only Windows extensions (Creator / Type is not used on Mac anymore).
SR RELATIONS	Sets compatibility mode on (in SuperReport Pro 3 it is a persistent property of the report, not a global one).
SR Set Script Callback	When used with area = 0 sets compatibility mode on (in SuperReport Pro 3 it is a persistent property of report, not a global one).
SR SetPrinter	Does not check for valid printer name.
SR Print Disk & SR Print HTML	Use SR_Export or its variants.
SR Power Menu	Not implemented.
SR Package	Returns 0 and invokes the debugger if this has been enabled.
SR Get Free Memory	Not implemented.
SR SWAP HANDLES	Not implemented.

Registering SuperReport Pro

The SR_Register command takes just one license key, and it returns 0 if successful and an integer between 1 and 12 if not OK. There is a list of the registration error codes in the command description.

What's New in SuperReport Pro Version 3

Adjust Object Size by Style

A new menu item **Object>Adjust Object Size by Style** computes an object's height according to the style specified for that object. In SuperReport 2 when you resized a text, variable, or field object, the height was adjusted to show full line(s)); in SuperReport Pro 3 you have to do it manually. The keyboard shortcut is shift-double-click.

Count Pages

A count pages option (SRP_Report_CountPages) has been added, so now you can use "Page 1 of 9" for example (however, note that the report is processed three times).

Dynamic Text

Text objects can use "<% [+] [=] variable [; format] %>" when SRP_Text_Dynamic = 1

Group object

Objects can now be grouped together by selecting them and clicking on the **Group Object** tool in the toolbar.

HTML Export

The HTML export has been improved.

Multiple headers/footers

A multiple headers/footers option has been added (first page/second+ pages/last page).

To function properly on the last page, SRP_Report_CountPages must be set to 1 (if count pages is 0, it will never print).

Native drawing of Text

SuperReport Pro uses CoreText on MacOS and GDI+ on Windows. This restricts the version of OS that SuperReport Pro can run on: SuperReport Pro version 3.x requires MacOS 10.5 or higher and Windows XP SP2 or newer.

Only fonts and font faces supported by these technologies can be used in SuperReport Pro.

In particular, GDI+ does not support non-TrueType fonts on some Windows versions.

GDI+ (Windows) does not support OTF fonts.

Multistyle (attributed) text

SuperReport Pro supports the multistyled text feature of 4D v12 and above.

Object Properties

Objects have new properties - name, id - which can be used in XML/HTML exports.

Scripts

Scripts now support flow control structures (but you must not use local variables).

Scripts are stored tokenized, eliminating problems between various localized 4D versions (e.g. the English version no longer fails to run the French demo due to the command names).

Table object

A new object has been added to the toolbar: the Table Tool. Use this to create tables within your report.

Unicode

SuperReport Pro supports Unicode.

Watermark section

A Watermark section has been added. The watermark will be printed on every page, either underneath or on top of all other printing.

XML

An area's settings can be saved as XML into a variable or field. Note that if 4D is run in non-Unicode mode, the size of any text is restricted to 32k characters. This may be enough as long as you don't have any large pictures in the report, however, it is recommended that you use the application in Unicode mode.

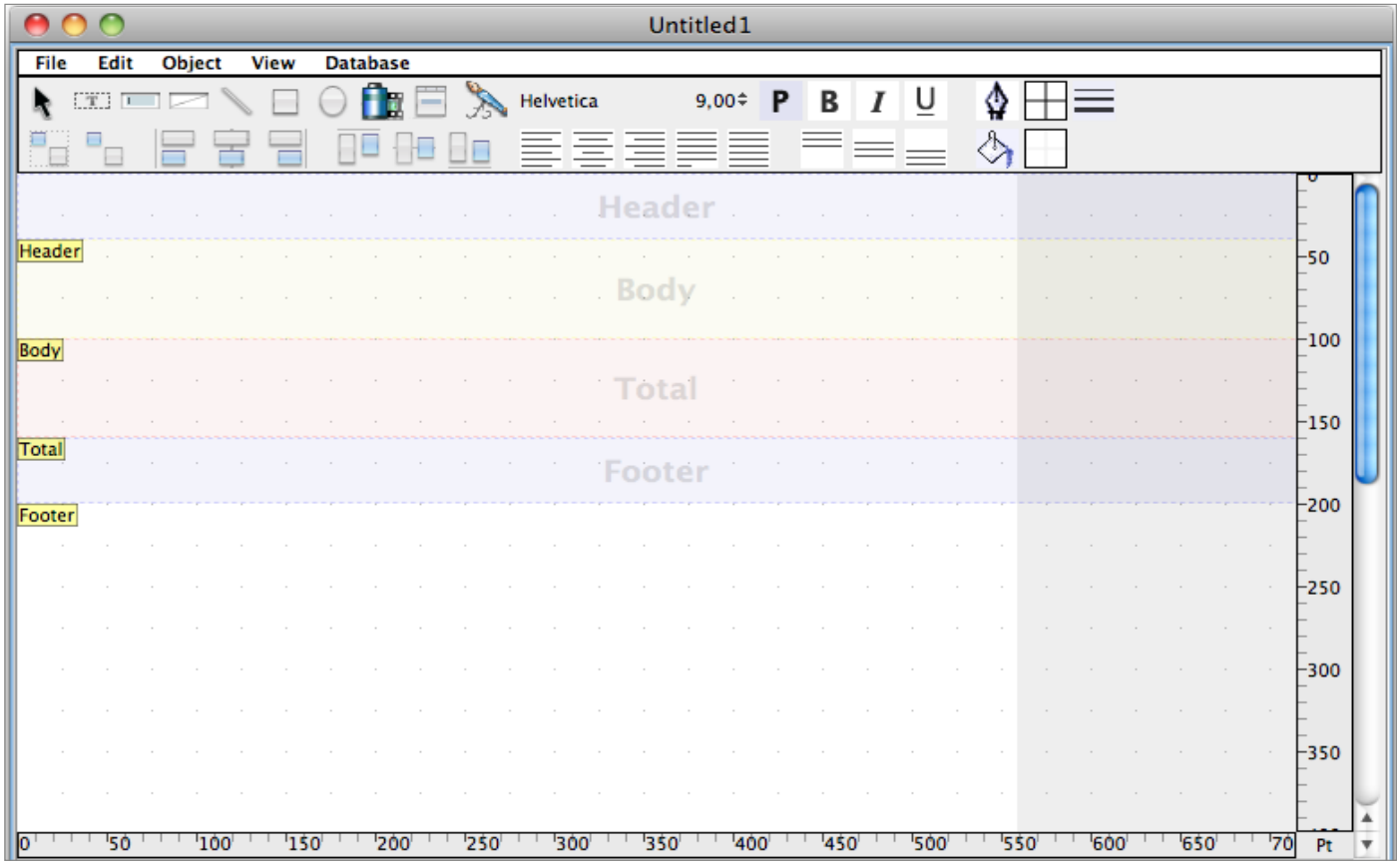
Use the old commands for loading/saving reports (using blobs) if you have to run in non-Unicode mode.

Unlimited Undo/Redo support

SuperReport Pro supports unlimited undos and redos for all user actions (to be precise, up to 65536 possible consecutive actions).

Other Improvements

You'll also notice many additional little improvements and new features such as "center object on page", "bind object to bottom of a group/section", "print break footer at bottom of page, not below body", and other goodies.



Programming SuperReport Pro

SuperReport Pro Objects

There are several object types, called "Object kinds" and several ways to identify a SuperReport Pro object.

Object kinds and their respective properties are accessed through Property Constants, e.g. specific properties for objet kind “Line” are available as Line Properties.

Object Kinds

Here is the list of all possible object kinds, accessed through the SRP_Object_Kind text property:

- SRP_ObjectKind_Style

Styles can be predefined for various object kinds, including text attributes, colors, rotation, etc.

- SRP_ObjectKind_Area

A SuperReport Pro plugin area, in a 4D form or an external window.

- SRP_ObjectKind_Report

The SuperReport Pro report object itself, in a 4D form, an external window or offscreen.

- SRP_ObjectKind_Section

A section of the report. Sections include Header, Body, Break Header, Break Footer (where Total is a Break Footer with Break Level = 0), Footer and Watermark.

- SRP_ObjectKind_Group

A group of objects.

Static objects

- SRP_ObjectKind_Line
- SRP_ObjectKind_Oval
- SRP_ObjectKind_Rectangle
- SRP_ObjectKind_Picture
- SRP_ObjectKind_Text (not fully static, since values can be embedded as <%variable%>)

4D objects

- SRP_ObjectKind_Variable
- SRP_ObjectKind_Field

Table objects

- SRP_ObjectKind_Table
- SRP_ObjectKind_Header
- SRP_ObjectKind_Column
- SRP_ObjectKind_Footer
- SRP_ObjectKind_Guide

The guide lines displayed in the SuperReport Pro design editor.

- SRP_ObjectKind_DataSource

Source of data (variables, records or fields). The Data Source Properties include information regarding the interface to the 4D database, such as the variable name, relate one / relate many, SuperReport Pro callback scripts, etc.

Area / Report Identification

Many SuperReport Pro commands expect a reference as their first parameter:

SR_NewObjectFromXML

(areaReportRef:L; objNum:L; XML:T; parent:L) → error:L

This reference is called “Area / Report reference” and specifies where to apply the command, i.e. which SuperReport Pro report displayed in a window (4D form or external), or offscreen is involved.

It is documented as “areaReportRef” in the Command description and as $areaRepRef in examples.

Depending on the context (as explained below), this value can either be the Report reference, the Area reference or the Printed report reference.

Report reference

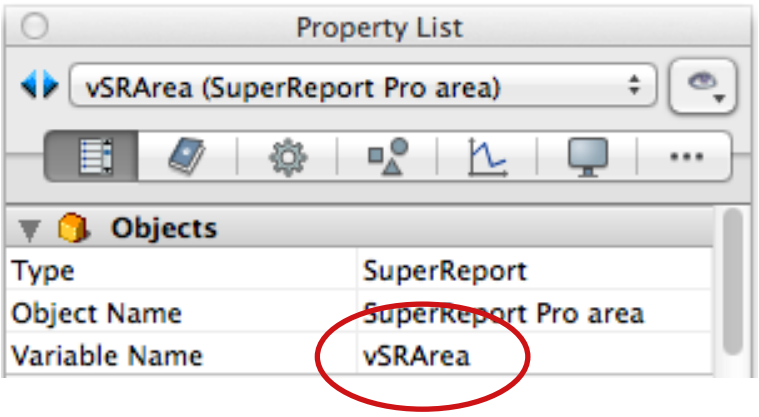
For offscreen areas, this value is returned by the SR_NewReport or SR_NewReportBLOB commands used to create the report.

It uniquely references the report being edited (programmatically since it is an offscreen).

This reference also exists in edited on screen areas (use the SRP_Area_Report property), but you don't really need it since this type of area has an Area reference, which should be used instead.

Area reference

This reference always exists in edited on screen areas: it is the value of the 4D variable assigned to the Plugin area in design mode.



Simply pass this variable as your first parameter to the commands that expect an Area / Report reference.

Note: the Area reference is unusable on 64-bit servers but who wants to edit a report on a server anyway?

Printed report reference

This is the internal reference of the report being printed or exported. This number is temporary and only valid during printing or exporting.

It is not to be confused with the Report and Area references used to refer to the report itself when building it programmatically or in the design editor.

The internal SuperReport Pro variable SRArea used in any report script contains this value (the name of the variable can be modified through the SRP_Area_VarArea property).

Which Area / Report reference to use?

All commands accept both Area and Report references.

SR_GetXXXProperty / ***SR_SetXXXProperty*** work with commands only with Variable, Field, Table and Table Column object kinds during printing or exporting (i.e. in SuperReport Pro scripts) therefore potentially accept all three references: Area, Report, Printed report (SRArea).

- Always use the Printed report reference when print / export is in progress: pass SRArea to the commands that you are calling in a script, or in a 4D method called by a script
- Use the Area reference for an edited on screen area: pass the plugin area 4D variable
- Use the Report reference for offscreen reports when modifying them programmatically: pass the value that was returned in the first parameter of SR_NewReport or SR_NewReportBLOB

Object Identification

Object number

This long integer value is assigned internally by SuperReport Pro and uniquely references any object from the report, including the report itself (which always has the value 1 as an object - not to be confused with the Area / Report reference).

Note that this number, although always unique, is not static. It may change on the fly as the report definition is modified.

Here is the technical explanation: initially the objects are numbered sequentially, but the order can change at any time, e.g. removing an object, including ungrouping a group object (which deletes the group as an object).

Internally, the Object number is the index to the objects array. This is a single array for all objects in the report, thus the position is not modified when you move objects, group them or change their print order. It does change, however, when an object is deleted.

Therefore, when you get an array of objects and you manipulate one object having the effect of renumbering others, you must re-fetch the array.

When used in commands, where appropriate:

- 0 = selected objects
- 1 = the report itself
- Any other number = a specific object

This number is documented as “objNum” in the Command description. and $objNum in the examples.

Object ID

This is a text value for most objects, except Styles, which have a long integer ID.

The value is either set by you as the developer, or automatically by SuperReport Pro for long integer style IDs in the design editor.

For other object kinds, when a new object is created in the editor, the ID is set to the (localized) object kind and a sequence number, e.g. "Text_4". Otherwise the default ID is an empty string.

The Object ID is accessed through the SRP_Object_ID property.

Note: this value is not unique. You can set the same ID to different objects, it is up to you to check the uniqueness as needed. See for example Warning: Style ID Uniqueness in the Properties by Theme section.

Duplicate values can be used e.g. for XML formatting (like using the ID as a class ID for those object-minded programmers out there).

The Object ID is generally used only by XML / HTML export. The only exception is Style (long integer ID), where text objects (and tables) can use a style from the Styles set by ID.

To get the Object ID from the Object number:

SR_GetTextProperty ($areaRepRef;$objNum; SRP_Object_ID)

To get the Object number from the Object ID (see also the SR_GetObjectsByPropertyValue command):

$error:=***SR_FindObjectByID***($areaRepRef;$objectID;$objNum)

Because the Object ID is not necessarily unique, it will return the first Object number having the requested ID.

Object name

This is a text value, available for many objects, set by you as the developer.

When a new object is created in the editor, the Object name defaults to an empty string, except for Styles, where names are initialized to “New Style” when added in this context.

The Object name is accessed through the SRP_Object_Name property.

Note: this value is not unique. You can set the same Object name to different objects, it is up to you to check the uniqueness as needed.

To get the Object name from the Object number:

```
$objectName:=SR_GetTextProperty ($areaRepRef;$objNum; SRP_Object_Name)
```

Object print reference

This is the internal reference of each object being printed or exported. These numbers are temporary and only valid during printing or exporting.

They are not to be confused with the Object number used to reference objects in the report itself when building it programmatically or in the design editor.

The internal SuperReport Pro variable SRObjectPrintRef used in the object script (not in Start/Body/End or Section scripts) contains this value (the name of the variable can be modified through the SRP_Area_VarObject property). SRObjectPrintRef is a 4D variable for use in the object script, analog to the 4D **Self** pointer.

Note: the SRObjectPrintRef variable replaces the old SRObjectID, which is maintained for compatibility. See SuperReport Pro variables.

Similarly, as previously described the internal SuperReport Pro variable SRArea contains the Printed report reference of the report being printed or exported.

In other words, SRObjectPrintRef (object while printing/exporting) is to the Object number (object in edited or modified report) what SRArea (Printed report reference) is to the Area reference or Report reference (edited or modified report).

To get the Object name from the Object print reference while printing or exporting:

```
SR_GetTextProperty (SRArea;SRObjectPrintRef; SRP_Object_Name)
```

Note: printed (or exported) reports support the ***SR_GetXXXProperty***/***SR_SetXXXProperty*** commands only with the following object kinds: Variable, Field, Table and Table Column. Use the Printed report reference and the Object print reference in this context.

Commands and Functions

SuperReport Pro has its own collection of commands and functions that you use to control your SuperReport Pro areas, to find out what actions the user has taken, and to do whatever processing is needed as a result.

Properties

Each command theme has its own set of properties that can be used to get or set various aspects of the report, and for each property a 4D constant has been defined.

You'll find a complete reference in the Properties by Theme section.

Commands

The commands are organised into themes which relate to a particular part of the SuperReport Pro functionality: Access, Getters, Setters, Objects, Printing and Miscellaneous.

The "Getter" and "Setter" commands are used to set the various properties of the report, and to find out what the specific settings are, or what action the user has taken.

For example, you can find out the name of a specified object on a report with the ***SR_GetPtrProperty*** "getter":

```
$error:=SR_GetPtrProperty ($areaRepRef;$objNum;SRP_Object_Name;->$objName)
```

A SuperReport Pro command syntax looks like this:

SR_SetLongProperty

(areaReportRef:L; objNum:L;property:T; value:L)

areaReportRef is the SuperReport Pro Area / Report reference.

objNum is the Object number you want to access.

property is a constant that tells SuperReport Pro exactly what information you want to set (see the Properties by Theme, Appendix 4 and Appendix 5 sections to find out what the possible constants are).

Each parameter is followed by a colon and a letter indicating the type of data required for that parameter:

:L = longint

:O - blob

:P - picture

:R = real

:T = text

:Z = pointer

Array types are prefixed with an "**A**": **AT** stands for text array.

Note that Boolean values are passed or returned as longints, where 1 = true and 0 = false.

Each is preceded by one of three arrow signs, which indicate whether it is a value that you pass to the command or one that the command returns to you, or a value that is passed, then modified and returned by the command in the same parameter:

➞ parameter A value that you pass to the command

➟ parameter A value that is returned by the command

Note: when calling a plugin command, all omitted parameters are initialized to the NULL of the respective types (0, 0.0, "", !00:00:00!, …).

You can find complete descriptions of the commands, along with examples, in the Command Reference section, and descriptions of all the properties in the Properties by Theme section.

The Properties by Theme section includes details of how to use each property; the Type column tells you what type of data it requires, and this is matched to the command variant.

Functions

Functions return a result code when they are called. Usually this will be the information you requested.

Their syntax looks like this:

SR_GetLongProperty

(areaReportRef:L; objNum:L; property:T) → value:L

areaReportRef is the SuperReport Pro Area / Report reference.

objNum is the Object number you want to access.

For example, you can get all sections from a report using

```
$error:=SR_GetObjects ($areaRepRef;1;SRP_ReportSections;$objectNums)
```

and then access the first section's name using

```
$objectName:=SR_GetTextProperty ($areaRepRef;$objectNums{1};SRP_Object_Name)
```

property is a constant that tells SuperReport Pro exactly what information you want to get (see the Properties by Theme, Appendix 4 and Appendix 5 sections to find out what the possible constants are).

value is the result that the function returns.

For example, suppose we want to find out which tool the user selected. We can use the ***SR_GetLongProperty*** "getter" function to find out:

```
C_LONGINT($toolSelected)
$toolSelected:=SR_GetLongProperty ($areaRepRef;1;SRP_Area_Tool)
```

Getters and Setters

Most of the commands are either “getters” or “setters”: they either **get** information about a specific property, or they **set** a specific property.

The Getters and Setters are each available in five variants, which allow for the different data types of the properties: Long integer, Pointer, Real, Text, and Text Arrays.

For example, to set the “show ruler” property for a report, you use the SR_SetLongProperty command along with the SRP_Report_ShowRuler property:

```
SR_SetLongProperty ($areaRepRef;1;SRP_Report_ShowRuler;1)
```

The pointer options (SR_GetPtrProperty and SR_SetPtrProperty) must be used to manipulate pictures and blobs.

Also, they enable you to use just one version of the command for getting and setting all the relevant properties; you pass a pointer to the variable instead of the actual value.

When to use the commands and functions

Most SuperReport Pro commands and functions need to be passed an Area / Report reference to the report on which they will act.

Since SuperReport Pro areas are initialised in the On Load phase of a layout, the commands must be called during this phase or afterwards; if you try to call any SuperReport Pro commands before the form has been loaded, you’ll get an error message because 4D does not recognise the area reference.

Anatomy of a SuperReport Pro Command

Each command you write must adhere to a specific syntax in order for it to be correctly understood by SuperReport Pro. Some commands (the “getters” and the pointer variants) return a result code: these are functions. See the Command Reference section for the requirements for each command.

You can check the result code to find out if a function executed OK or if there was a problem and, if so, get some information about what that problem was.

For example, ***SR_GetProperties*** is a function which returns information about all properties of the referenced report or object into two or, optionally, three text arrays:

```
ARRAY TEXT($propertyIDs;0)
ARRAY TEXT($propertyValues;0)
ARRAY TEXT($propertyNames;0)  // (this is the optional one)
$error:=SR_GetProperties ($areaRepRef;0; $propertyIDs;$propertyValues;$propertyNames)
```

If the function executed successfully, $error will be 0; if not, $error will contain an error number. You can check the meanings of the error codes in the Result Codes list.

Every command consists of the command name followed by two or more parameters. The first parameter is always a reference to the SuperReport Pro area. For example, the SR_SetRealProperty command sets a specific property that requires a real number:

```
SR_SetRealProperty ($areaRepRef;1;SRP_Report_Zoom;$zoomRatio)
```

This command sets the scaling for the SuperReport Pro report in $areaRepRef to the value specified in the $zoomRatio variable.

Commands that get or set properties for an object all include the property that you want to affect, and a value to use to specify an option (if it's a "setter") or to receive the result (if it's a "getter").

All SuperReport Pro commands are described in the Command Reference section along with examples of how to use them.

Debugger

When the SRP_Area_TraceOnError property's bit 0 is set to true in interpreted mode (the default), if there is an error in a command that does not return an error code, and you are using 4D in interpreted mode, the 4D debugger window will automatically open with the line immediately following the problem line highlighted.

In compiled mode, if SRP_Area_TraceOnError property's bit 1 is set to true an alert is displayed with the error code, the SuperReport Pro command, the calling 4D method and the property used (selector, see Property Values, Constants and XML Names).

Customizing the SuperReport Pro Area

There are a variety of ways you can customize the SuperReport Pro area to suit your application's needs.

For example, you may want to disable access to the Scripting option for end-users, or hide the menu altogether. These options are controlled through the use of the "setter" commands and the appropriate properties.

We will give a few examples below; you can find full details of the available properties in the Properties by Theme section.

In your form's On Load event, call one or more of the Setter commands with the properties you want to set.

The following example will hide SuperReport Pro's ruler:

```
SR_SetLongProperty ($areaRepRef;1;SRP_Report_ShowRuler;0)
```

The following example will disable the zoom window:

```
SR_SetLongProperty ($areaRepRef;1;SRP_Report_ShowZoom;0)
```

This example will set the report's zoom to 50%:

```
SR_SetRealProperty ($areaRepRef;1;SRP_Report_Zoom;.5)
```

SuperReport Pro Variables

There are a number of internal variables used by SuperReport Pro. Following is a description of each variable and their usage during reporting operations.

In addition, most of these variables can be used in any script which is executed by SuperReport Pro during print or export (HTML or other). This is the “Scope column”:

- **SRP** means that it is not a 4D variable, therefore cannot be used in a script or a 4D method. However these values can be included in the report as variable objects or embedded into a text inside variable placeholders (<%variable%>)
- **4D** means that, in addition to the above, these variables can be accessed as regular 4D process variables in SuperReport Pro scripts or 4D methods

The variable names below can be accessed (including modified) through the SRP_Area_VarXXX properties (“Property for name” column):

Variable name	Type	Scope	Property for name	Description
SRArea	**C_LONGINT**	4D	SRP_Area_VarArea	Current Printed report reference.
SRBegHTML	**C_TEXT**	4D	SRP_Area_VarBegHTML	HTML “header”. See Exporting to HTML.
SRDate	**C_DATE**	4D	SRP_Area_VarDate	Contains the current date.
SRDateTime	**C_LONGINT**	SRP	SRP_Area_VarDateTime	Contains the current date and time. See note below.
SREndHTML	**C_TEXT**	4D	SRP_Area_VarEndHTML	HTML “footer”. See Exporting to HTML.
SRName	**C_TEXT**	SRP	SRP_Area_VarName	Contains the SRP_Object_Name of the report being printed or exported.
SRObjectPrintRef	**C_LONGINT**	4D	SRP_Area_VarObject	Contains the current Object print reference for which you have a script attached. See note below.
SRPrintSection	**C_LONGINT**	4D	SRP_Area_VarPrintSection	Set to 1 (print - default value) or to 0 (don't print) in the section - or any object - script to include the current section in the print/export or to skip it.
SRPage	**C_LONGINT**	4D	SRP_Area_VarPage	Contains the current page.
SRPages	**C_LONGINT**	4D	SRP_Area_VarPages	Contains the printed report total page count.
SRRecord	**C_LONGINT**	4D	SRP_Area_VarRecord	Contains the current record number or iteration value.
SRTime	**C_LONGINT**	4D	SRP_Area_VarTime	Contains the current time.
SRCurrentRun	**C_LONGINT**	4D	SRP_Area_VarCurrentRun	Possible values 1, 2, 3. When using “calculate number of pages”, the report is processed three times. If you need to use Request or **QUERY**, check for SRCurrentRun=1 to execute it only the first time.
SRRepeatNum	**C_LONGINT**	4D	SRP_Area_VarRepeat	Only valid during execution of a script associated with a repeating object. When using repeating objects, the script can know which iteration of the repeating object has to be handled (especially when using arrays).

Additional notes

- The SRObjectPrintRef variable replaces the old SRObjectID, which name was misleading since it refers to the Object print reference, not to the Object ID. SRObjectID is maintained for compatibility, but only the name SRObjectPrintRef can be modified with SRP_Area_VarObject.
- Longint, time and date values can be embedded with a standard 4D format e.g. <%SRDate;7%>
- Printing is the only valid context for SRPage and SRPages (not export)
- SRPages will default to -1 if page counting is not activated (SRP_Report_CountPages property set to zero or when exporting)
- Date and time variables are set upon print startup, therefore they will display the same value on all pages or instances within a given print or export session
- SRDateTime is limited to the SuperReport Pro scope (not a 4D variable). This long integer value is in UNIX time format (seconds since 1/1/1970). The default format is "%Y-%m-%d %H:%M:%S %Z".

Table objects

SuperReport Pro version 3 provides a unique, powerful feature: table objects.

Very similarly to AreaList Pro, SuperReport Pro table objects can display 4D arrays (including 2D arrays) or fields. Tables include columns with headers, body and footers. As a matter of fact AreaList Pro's tables can directly be printed using SuperReport Pro with the ***AL_SuperReport*** command.

Commands

You can also create and modify your own SuperReport Pro tables using the following commands:

- SR_NewObject (with the SRP_Table property)
- SR_GetObjects (with the SRP_TableHeaderRowMask, SRP_TableColumns and SRP_TableFooterRowMask properties)
- SR_ModifyTable

Object kinds

SuperReport Pro tables use the following object kinds:

- SRP_ObjectKind_Table
- SRP_ObjectKind_Header
- SRP_ObjectKind_Column
- SRP_ObjectKind_Footer

Table properties

The following properties are associated to SuperReport Pro tables:

- Table / Header / Column / Footer
- Table Properties
- Table Header Properties
- Table Column Properties
- Table Footer Properties

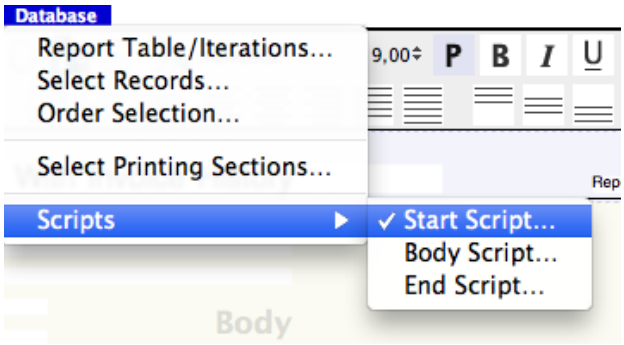
See for example Creating Reports Procedurally.

Extending SuperReport Pro with Scripts

You can add scripts to your SuperReport Pro reports and to objects on a report.

Execution Cycle scripts

Scripts can be added to the start of the report, the body, and the end of the report. To add one of these, choose **Scripts** from the **Database** menu, and then select one of the three sub-menu options:



The checkmark next to the **Start Script...** option in this example indicates that a Start Script has been added.

To abort printing from a SuperReport Pro script (Start or Body script, or any object script, see below):

SR_AbortPrinting

The above line can either be inserted into the script itself or in a 4D project method called by the script.

You can find other examples of using scripts in the Tutorial section of the User Guide.

Object Scripts

Like standard 4D reports created using the Form Editor, SuperReport Pro objects may contain scripts which may call any 4D command, method, or plug-in command. Using SuperReport Pro scripts, you can further customize your reports to produce output which cannot be created using standard SuperReport Pro objects.

For example, let's say you wanted to create a report with a listing of sales, broken down by customer, and provide an average sale for each customer as well as an overall invoice average. While SuperReport Pro can handle most of these tasks for you, utilizing the advanced break processing capabilities built-in to SuperReport Pro, the extended averaging requirements would not be possible without the support of object scripts.

Using SuperReport Pro's object scripts, you can place custom code on just about any SuperReport Pro object, providing a very complete and flexible reporting system. Scripts can be attached to the following object kinds: Datasource (the Report's Start / Body / End scripts), Section, Variable, Field, Table and Table Column.

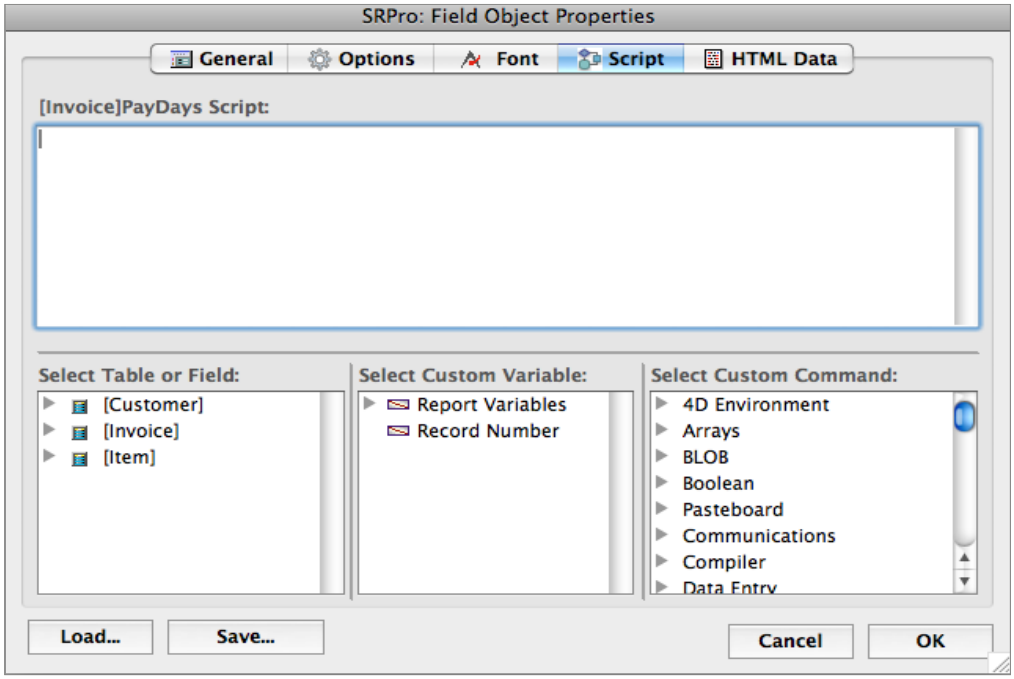
Note: printed (or exported) reports support the ***SR_GetXXXProperty*** / ***SR_SetXXXProperty*** commands only with the following object kinds: Variable, Field, Table and Table Column. Use the Printed report reference and the Object print reference in this context.

For example, programming a variable object's script to display its value in bold red if negative, using the SRArea and SRObjectPrintRef standard variable names:

```
If (myVariable<0)  //negative number
   SR_SetTextProperty (SRArea;SRObjectPrintRef;SRP_Style_TextColor;"red")
   SR_SetLongProperty (SRArea;SRObjectPrintRef;SRP_Style_Bold;1)
End if
```

To add a script to an object, double-click on the object. The object properties dialog will open; the exact contents of the dialog vary depending upon the type of object, but they will all have a Script option.

For example, the **Field Properties** dialog looks like this, with the Script tab selected:



Note: option double-click on a scriptable object will directly open the object script editor.

SuperReport Pro Script Limitations

While SuperReport Pro provides the ability to easily create custom scripts, there are a few limitations that you should be aware of when creating your scripts.

1. SuperReport Pro scripts are not executed in compiled form, regardless of your application's compiled status. While this typically won't be a noticeable issue, you should be aware that scripts are slower than compiled code.
2. All scripts are executed in the same fashion as if the 4D **EXECUTE FORMULA** command had been called.
3. When selecting a table or field which has been customized (using SuperReport Pro's structure customization feature), the script editor will display the actual table/field information as configured at the structure level.
4. You must not use local variables ($var) in compiled mode.
5. You cannot manipulate sections or report in a script, or remove/add objects. You can only modify existing objects in this context.
6. You use SQL code with **Begin SQL**/**End SQL** statements.

Understanding the SuperReport Pro Event Cycle

Due to the extensive reporting features offered by SuperReport Pro, there is the potential for some level of confusion for even the seasoned 4D programmer. The following information outlines the standard SuperReport Pro Event Cycle.

SuperReport Pro generates reports in the following sequence (in pseudo-code):

```
Execute StartProcedure
Determine how many iterations are required
While (Current Iteration < Maximum Iterations) do
   Execute BodyProcedure
   If (Break Values have changed)
      Process appropriate Break Header sections
      Process appropriate Break Footer sections
   End if
   Process Body section
End while
Process Total section
Execute EndProcedure
```

Section Processing

In processing any section, the following sequence of events takes place:

```
Execute section script
For (i = 1 to number of objects in this section) do
   Execute object script
   Get object (field / variable / array) value
End for
```

Dealing with multi-platform Issues

Fortunately, SuperReport Pro does a good job of conditionally handling most multi-platform issues. However, there are some cases when some additional developer control is necessary to support multi-platform issues:

- Dealing with platform pathnames
- Dealing with Print Drivers

Dealing with platform pathnames

All MacOS files use a directory delimiter of ":" (ASCII 58), while Windows directories are delimited with a "\" character. One approach to dealing with these differences is to use a 4D variable, which contains the delimiter.

```
$delim:=Get 4D folder≤Length(Get 4D folder)≥
```

Then you could use the 4D function **Application file** or **Structure file** to return the pathname to either the application or structure file as a starting point to determine the pathname to a "Reports" directory.

For some useful routines for obtaining the parent pathname for an application or structure file, refer to the 4D Language Reference.

Dealing with Print Drivers

Due the large number of print drivers available, configuring reports for most printers can be a very difficult task. Following is a list of some tips which can make designing your reports a manageable job.

1. Try to leave at least 1/4" margin on the edges of your reports to account for a wide variety of print drivers
2. Stick with common fonts.
3. Be careful when using enhanced font formatting attributes such as Bold, Italic, Underline, etc. Make sure that your print driver knows how to deal with multiple styles. Also MacOS does not support "synthetic" styles: you can only print those typefaces that are available in your system fonts.
4. Test, test, test! Try to test your reports on as many different printers as possible… you would be surprised at the differences between drivers.

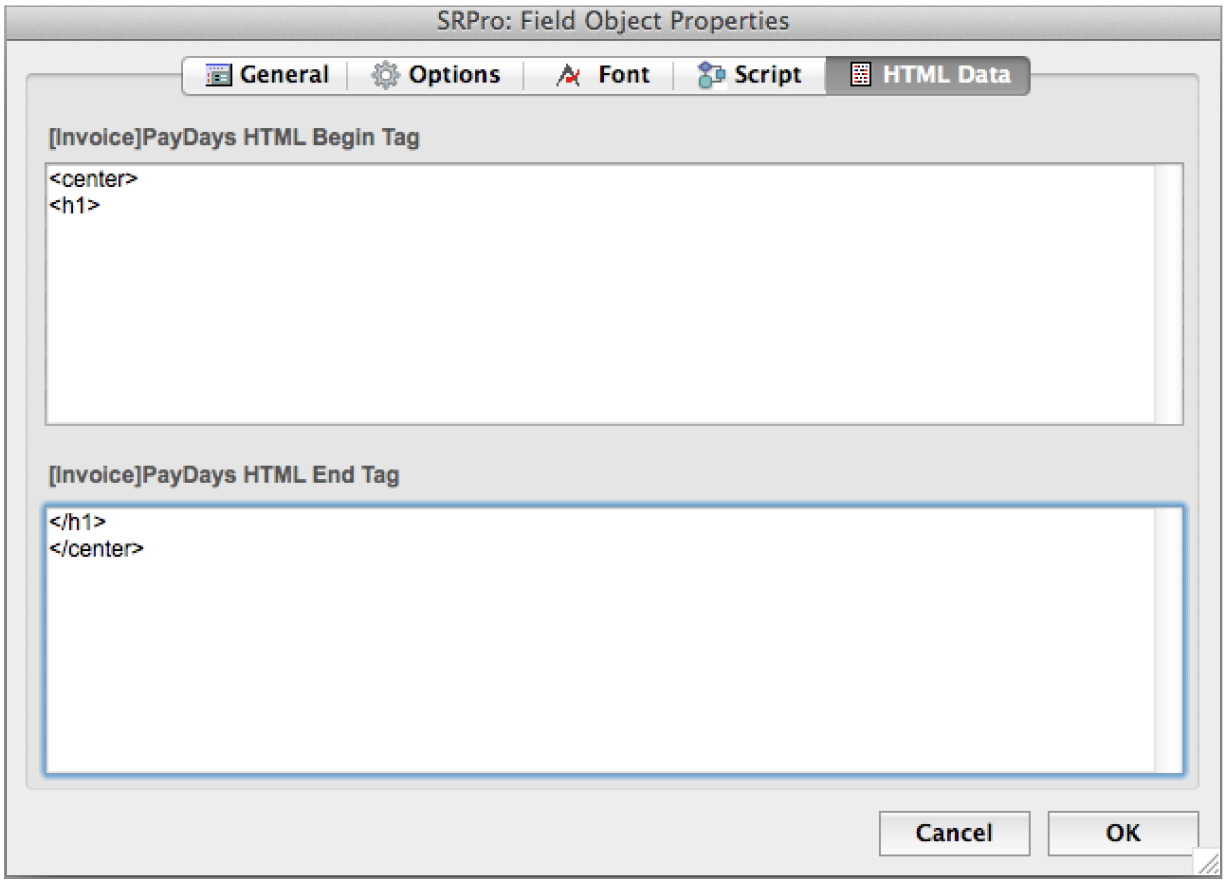
If you have a tip that would be useful to other SuperReport Pro users, please let us know so that we can post this information in future documentation and release notes, as well as on our web site.

Send tips to the SuperReport Pro forum.

HTML Support

SuperReport Pro provides the ability to create standard HTML reports.

Each SuperReport Pro object may contain custom HTML tags, which are inserted before and after the corresponding data object is sent to the HTML content when exporting the report in HTML format.



Using the HTML Data Tab, you can supply the following items:

- Begin Tag

The begin tag information is sent to the HTML content before the corresponding data is sent.

- End Tag

The end tag information is sent to the HTML content after the corresponding data is sent.

These values are specific to the edited object. Each set of HTML Begin / HTML End Tags is used with each respective object every time it is exported.

A general set of begin / end tags for the report itself is available as standard SuperReport Pro SRBegHTML and SREndHTML variables (see below, Exporting to HTML):

```
SRBegHTML:="<html><body bgcolor='#ffffff'>"  //sample start file tag
SREndHTML:="</body></html>"  //sample end file tag
```

Using Custom Tag Variables

In addition to entering static HTML tags in the HTML Dialog, you can also embed custom HTML tag variables, which are actually 4D variables which may contain any valid HTML tag (or group of tags).

For example, if you have a set of HTML tags which you insert at the beginning of some (or all) of your report objects, you could place all the HTML data into a 4D variable, then reference the variable from within SuperReport Pro by using the custom variable tags.

To use a 4D variable within your HTML tags, enclose the variable with “<%” and “%>” markers. When the report is printed, SuperReport Pro will take the data between these markers, retrieve the data from the actual variable and insert it into the HTML stream as the file is created.

Using a 4D variable instead of static HTML tags in each object allows modifying all the HTML tags at once: for example, use "<%tMyHTMLPrefix%>" as the HTML Begin Tag in all relevant objects then assign the desired value to tMyHTMLPrefix.

Following is an example of a 4D variable that is inserted using the variable tags:

<h1>" or "<%vH1%>" with 4D variable vH1:="<h1>" will produce "<h1>" in the HTML stream

Note: the example above stands for non-attributed text, see Embedding variables in HTML export text.

The SuperReport Pro Execution Cycle

When creating HTML reports, it is important that you understand the 4D execution cycle as this is how you will determine when specific tags should be included so that your HTML files will be correctly structured.

For more information on the SuperReport Pro Execution Cycle, please refer to the Understanding the SuperReport Pro Event Cycle section earlier in this chapter.

Exporting to HTML

To export to HTML, use SR_Export, SR_ExportBLOB, SR_ExportIntoBLOB or SR_ExportBLOBIntoBLOB.

Specify either SRP_Export_XML or SRP_Export_HTML (among other flags) in the second parameter.

Note: Lines, Rectangles and Ovals are not exported.

At the start of processing, the "header" is written:

- If the SRBegHTML variable exists and is not empty, it is written as-is; otherwise, the standard prolog is written (declarations + <html><head><title>report name</title></head><body>).

At the end, the "footer" is written:

- If the SREndHTML variable exists and is not empty, it is written as-is; otherwise, the standard epilog is written (</body></html>).

Sections are written as follows:

- If the SRP_Object_HTMLPrefix property exists (is not empty), it is written as is; otherwise, a simple <DIV> is written.
- All objects are exported.
- If the SRP_Object_HTMLSuffix property exists (is not empty), it is written as-is, otherwise a simple </DIV> is written.

Objects are written as follows:

- If the HTML prefix or suffix property is not empty, the prefix is written as-is, the object is written, the suffix is written; otherwise a simple is written using the name and id attributes of the respective object, the object is written.
- Table objects are exported as HTML tables using the tags <thead><th><td> for headers and <tbody><tr><td> for data.
- Picture object exports the SRP_Picture_HTMLReplace property instead of the actual data.
- How the respective objects are written depends on the SRP_Text_Attributed property of the object and the SRP_Export_PlainText bit in the **Options** parameter of the export command.

HTML prefix refers to SRP_Object_HTMLPrefix.

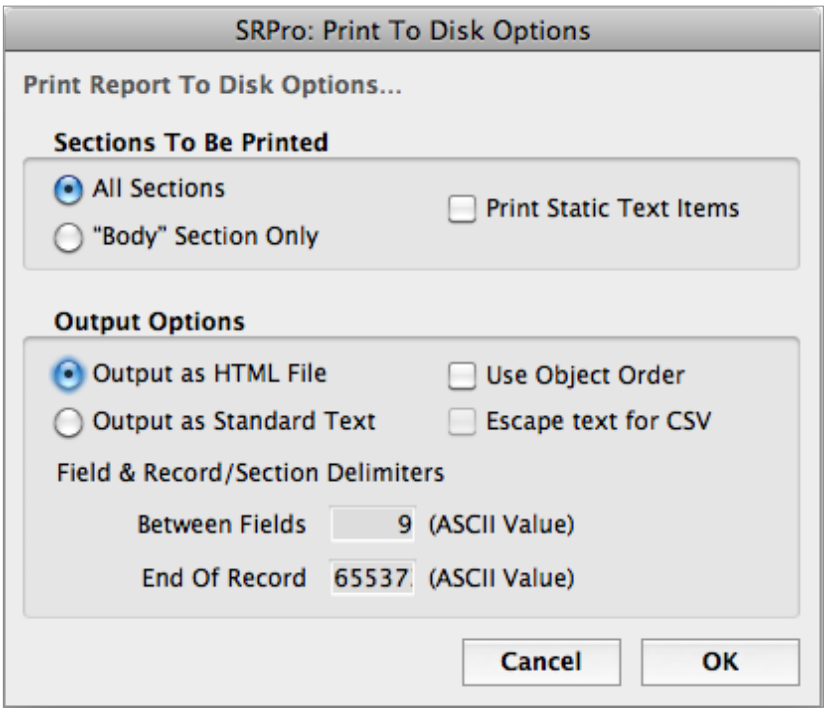
The HTML suffix means SRP_Object_HTMLSuffix.

Note: the font is not exported (SuperReport Pro 2.5 used <font face="xxx">).

There are three ways to export your reports in HTML format. The first method uses the SuperReport Pro Editor.

1. Select **Export**… from the **File** menu.

2. Select **Output As HTML File** in the **Output Options** section:



The second method is to use the SR_Export command with the SRP_Export_HTML option - for example:

```
$error:=SR_Export ($areaRepRef;SRP_Export_HTML | SRP_Export_Body | SRP_Export_Breaks | \
   SRP_Export_Total | SRP_Export_Headers | SRP_Export_StaticText;9;"full path to a file to create.html";13)
```

The third method is to use the SR_ExportIntoBLOB command - for example:

```
C_BLOB($reportBlob)
SET BLOB SIZE($reportBlob;0)
$error:=SR_ExportIntoBLOB ($areaRepRef;SRP_Export_HTML | SRP_Export_Body;9;$reportBlob;13)
```

In addition:

- SR_ExportBLOB is the same as ***SR_Export***, except that the source is a blob.
- SR_ExportBLOBIntoBLOB is the same as ***SR_ExportBLOB***, except that both the source and the destination are blobs.

Note: exporting to XML is very similar to the HTML export, using <Report>, <section>, <variable>, <text>, <picture> and <group> tags with name and id attributes, ignoring HTML prefix/suffix properties.

Creating Reports Procedurally

It's possible to use SuperReport Pro commands to create entire reports procedurally.

The following example creates a report template that could be used for printing an AreaList Pro area using the ***AL_SuperReport*** command (see the next section on using SuperReport Pro to print AreaList Pro areas).

```
// Start by creating a new empty report:
// $areaRepRef is either an AreaList Pro plugin area or an offscreen SuperReport Pro area
C_LONGINT($error;$i)
C_LONGINT($areaRepRef;$objNum;$section)

$error:=SR_NewReport ($areaRepRef;"";0)
SR_SetTextProperty ($areaRepRef;1;SRP_Object_Name;"ALP")
SR_SetLongProperty ($areaRepRef;1;SRP_Report_PageWidth;595)  // A4
SR_SetLongProperty ($areaRepRef;1;SRP_Report_PageHeight;842)  // A4
SR_SetTextProperty ($areaRepRef;1;SRP_Report_Margins;"12.5;12.5;12.5;12.5")

// Modify the data source:
// $objNum is the object that you want to manipulate
$objNum:=SR_GetLongProperty ($areaRepRef;1;SRP_Report_DataSource)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_DataSource_Source;SR Iterations Fixed)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_DataSource_Iterations;1)

// Create two new styles:
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Style;0)
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Style;0)

// Modify styles - the default and the two added:
ARRAY LONGINT($aStyles;0)
$error:=SR_GetObjects ($areaRepRef;1;SRP_ReportStyleSet;$aStyles)
ARRAY LONGINT($aStyleIDs;Size of array($aStyles))

For ($i;1;Size of array($aStyles))
   $objNum:=$aStyles{$i}
   SR_SetTextProperty ($areaRepRef;$objNum;SRP_Style_FontName;"Helvetica")
   SR_SetRealProperty ($areaRepRef;$objNum;SRP_Style_Size;8)
   SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Wrap;1)
   SR_SetRealProperty ($areaRepRef;$objNum;SRP_Style_LineSpacing;1)
   $aStyleIDs{$i}:=SR_GetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID)  // should be 0, 1 & 2

   Case of
      : ($i=1)  // first is the Default
         SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"Default")
      : ($i=2)  // second will be used for headers
```

```
      SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"Headers")
      SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Bold;1)
   : ($i=3)  // third will be used for data columns
      SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"Columns")
End case
End for

// Create header section:
$error:=SR_NewObject ($areaRepRef;$section;SRP_Header;0)
SR_SetTextProperty ($areaRepRef;$section;SRP_Object_Name;"Header")
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_Height;40)
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_LabelPos;575)

// Create a Text object in the header - our title:
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Text;$section)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"Title")  // needed for AL_SuperReport
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosWidth;570)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosHeight;40)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_VariableSizeH;1)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_VariableSizeV;1)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Text_DrawEmpty;1)  // remove if empty
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID;0)  // this should not be needed
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Size;14)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_BaseLineShift;3)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Text_Data;"Some Title")
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_HorAlign;2)  // center on page

// Create the body section:
$error:=SR_NewObject ($areaRepRef;$section;SRP_Body;0)
SR_SetTextProperty ($areaRepRef;$section;SRP_Object_Name;"Body")
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_Height;40)
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_LabelPos;575)

// Create a Table object in the body - the table which will be filled by AL_SuperReport
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Table;$section)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosWidth;570)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosHeight;40)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Table_NumHeadings;1)
 // one line in header - this should not be needed
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Table_NumColumns;1)
 // one column - this should not be needed
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_Align;2)  // center on page
```

```
ARRAY LONGINT($aObjects;0)
$error:=SR_GetObjects ($areaRepRef;$objNum;String(1;SRP_TableHeaderRowMask);$aObjects)
  //get the first header line objects
SR_SetLongProperty ($areaRepRef;$aObjects{1};SRP_Object_StyleID;1)
$error:=SR_GetObjects ($areaRepRef;$objNum;SRP_TableColumns;$aObjects)  //get the columns
SR_SetLongProperty ($areaRepRef;$aObjects{1};SRP_Object_StyleID;2)

//Create the footer section
$error:=SR_NewObject ($areaRepRef;$section;SRP_Footer;0)
SR_SetTextProperty ($areaRepRef;$section;SRP_Object_Name;"Footer")
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_Height;13)
SR_SetLongProperty ($areaRepRef;$section;SRP_Section_LabelPos;575)

//Create a Variable object in the footer - Date:
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Variable;$section)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Variable_Source;"SRDate")
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosTop;3)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosWidth;200)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosHeight;10)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_VariableSizeV;1)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_Align;1)  //left on page
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID;0)  //this is the default

//Create a Variable object in the footer - Page
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Variable;$section)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Variable_Source;"SRPage")
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosLeft;470)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosTop;3)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosWidth;100)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_PosHeight;10)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_VariableSizeH;0x0001)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_VariableSizeV;1)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_Align;3)  //right on page
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID;0)  //this is the default

//Clear page setup (automatically created):
C_BLOB($blob)
$error:=SR_SetPtrProperty ($areaRepRef;SRP_Report_PageFormat;->$blob)  //MacOS
$error:=SR_SetPtrProperty ($areaRepRef;SRP_Report_PrintSettings;->$blob)
$error:=SR_SetPtrProperty ($areaRepRef;SRP_Report_DevMode;->$blob)  //Windows
$error:=SR_SetPtrProperty ($areaRepRef;SRP_Report_DeviceNames;->$blob)
```

```
// Get the XML and destroy the report
C_TEXT($XML)
$error:=SR_SaveReport ($areaRepRef;$XML;0)
$error:=SR_DeleteReport ($areaRepRef)

// Test it
C_LONGINT($areaRepRef)
$areaRepRef:=Open external window(100;100;800;800;Plain form window;"SRP";"%SuperReport")
$error:=SR_LoadReport ($areaRepRef;$XML;0)
```

Printing AreaList Pro Areas with SuperReport Pro

SuperReport Pro v3 can also be used in conjunction with AreaList Pro version 9.4 and above to print AreaList Pro areas.

How it works

AreaList Pro v9.4 and above allows printing or saving as HTML through SuperReport Pro v3. It only takes two lines of code to print an AreaList Pro area.

Additional options are available, such as automatic column width and use of SuperReport Pro style properties instead of the existing AreaList Pro area settings.

You can either use the built-in SuperReport Pro template to print an AreaList Pro area “on the fly” or create your own.

The AreaList Pro Demonstration database includes a “Print with SuperReport Pro” button in the **AreaList Pro > Configuration Options**… dialog. It also includes a SuperReport menu, allowing printing with the default template or a custom template, and editing / creating your own templates.

In addition, AreaList Pro v9.7 and above allow table footers printing with SuperReport Pro.

Command and property

Use the following command to print with SuperReport Pro:

AL_SuperReport

(areaRef:L; template:T; options:L; styleOptions:L; title:T) ➞ result:T

Parameter	Type	Description
➞ areaRef	longint	Reference of AreaList Pro object on layout.
➞ template	text	XML SuperReport template or full path to a XML template or empty to use AreaList Pro’s built-in template.
➞ options	longint	0 = use current columns widths; 1 = use automatic width.
➞ styleOptions	longint	Style properties that should **not** be overtaken by AreaList Pro - see constants in SuperReport Pro manual, Style Features.
➞ title	text	Optional text centered in the header.
← result	text	

Use the following AreaList Pro property with ***AL_GetAreaTextProperty*** to retrieve the default template in XML format:

Constant	Get	Set	Per	Type	Default	Min	Max	Comments
ALP_Area_SRPTableTemplate	✔			text				Get the SuperReport Pro template used for report creation (stored in Resources/TableReport.XML) as XML

Creating the report

Creating a XMLreport from an AreaList Pro area is performed by the ***AL_SuperReport*** command:

AL_SuperReport (AreaRef:L; Template:T; Options:L; StyleOptions:L; Title:T) → result:T

- **Template** can be an XML template or full path to an XML template or empty to use AreaList Pro's built-in SuperReport Pro template
- **Options**: 0 → use current columns widths; 1 → use automatic width
- **StyleOptions**: bit-field - which style properties should not be overtaken from AreaList Pro - see constants in SuperReport Pro Style Features
- **Title**: optional text centered in the header

If you want to use your own SuperReport Pro template:

- **Title** will replace any text in all text objects named "Title" in the first header (must not be grouped - only direct children of the header)
- in the body section, the first Table object will be filled with headers / columns / footers; the table must have exactly one column (used as the template for all printed columns)

Example

```
$reportXML:=AL_SuperReport ($alpAreaRef;"";1;SRP_Style_HasFontName | SRP_Style_HasFontSize;\
    "My first AreaList Pro area printed using SuperReport Pro")
```

The code above means: fill the template but don't use the font name and font size defined in AreaList Pro (use the one stored in the template default style), columns will be auto-sized by SuperReport Pro (because the fonts are different, the AreaList Pro widths must be ignored).

Then use SuperReport Pro to edit the resulting report, save it, export it as HTML or print it:

```
$result:=SR_Print ($reportXML;0;SRP_Print_DestinationPreview | SRP_Print_AskPageSetup; "";0;"")
```

Custom templates

AreaList Pro's built-in SuperReport Pro template is obtained by the ALP_Area_SRPTableTemplate property, which gets the SuperReport Pro template that will be used for report creation by ***AL_SuperReport*** (stored in Resources/TableReport.XML) as XML:

```
$tableReportTemplate:=AL_GetAreaTextProperty (0;ALP_Area_SRPTableTemplate)
    // get the built-in SRP template from ALP
```

Then in SuperReport Pro you can edit and save your own template anywhere (in the data file or a document) for future use with ***AL_SuperReport***, e.g.:

```
$error:=SR_LoadReport ($areaRepRef;$tableReportTemplate;0)
    // load the SRP report and display it, save the custom template somewhere with the File menu
```

Demonstration database code examples

Print with SuperReport Pro (default template)

```
C_TEXT($reportXML)
C_LONGINT($result)
$reportXML:=AL_SuperReport (eArea;"";0;0;vTitle)
$result:=SR_Print ($reportXML;0;SRP_Print_DestinationPreview | SRP_Print_AskPageSetup;"";0;"")
```

Print with SuperReport Pro (custom template)

```
C_TEXT($reportXML ;$path)
C_LONGINT($result)
$path:=Select document("";"XML";"Select a SuperReport Pro template for ALP";0)
$path:=Document  // we actually need the full path
If ($path#"")
   $reportXML:=AL_SuperReport (eArea;$path;0;0;vTitle)
   $result:=SR_Print ($reportXML;0;SRP_Print_DestinationPreview | SRP_Print_AskPageSetup;"";0;"")
End if
```

Editing a custom template

```
C_TEXT($tableReportTemplate)
C_LONGINT($srpError)
C_LONGINT($window)
$tableReportTemplate:=AL_GetAreaTextProperty (0;ALP_Area_SRPTableTemplate)
   // get the built-in SRP template from ALP
If ($tableReportTemplate#"")
   $window:=Open external window(100;100;800;800;Plain form window;\
      "SuperReport Pro template for AreaList Pro";"%SuperReport")  //open the window for editing
   $srpError:=SR_LoadReport ($window;$tableReportTemplate;0)
      // load the SuperReport Pro report and display it, save the custom template somewhere with the File menu
End if  // closing the window will prompt for save if modified
```

SuperReport Pro Text Style Tags

If the **Attributed** option has been set (SRP_Column_Attributed, SRP_Field_Attributed, SRP_Footer_Attributed, SRP_Header_Attributed, SRP_Text_Attributed, SRP_Variable_Attributed) special tags can also be used in any text contained in a SuperReport Pro area to display styled characters.

Note: the tags described below are exacty the same as in AreaList Pro.

These tags work just like HTML tags: <tag>styled text</tag>.

Style	Tag
Bold	<b>
Italic	<i>
Underline	<u> or <ins>
Strike-through	<del>
Set font size to # points	<s #>
Increase font size by # quarters (1/4) of current size	<s +#>
Decrease font size by # quarters (1/4) of current size	<s -#>
Set font by name	<f "font name"> (needs to be quoted if the name contains more than one word)
Set color (any format can be used, e.g. <c 0xFFFF0000> <c 1.0,0,0> <c P123> <c dark orange>)	<c color name>

4D v12 (and above) internal format for styled text is stored as <SPAN STYLE="style attributes">
where the style attributes used by SuperReport Pro are:

- font-family
- font-size
- font-weight (bold/normal)
- font-style (italic/normal)
- text-decoration (underline/ line-through/none)
- color (#RRGGBB).
- background color (#RRGGBB)

It is also possible to set the format as attributed, and specify the style attributes using the SRP_Variable_Attributed and SRP_Variable_Format properties.

This feature is usable with Variable, Field and Table Column types objects.

Example:

```
C_TEXT($format)
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Variable_Attributed;1) // the object is "attributed"
$format:= "<c  blue>+## ###</c>;<i><c  red>-## ###</c></i>;<s +1><c  green><b>ZERO</b></c></s>"
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Variable_Format;$format)
```

With the above settings:

- Positive numbers will be printed in blue roman characters with a plus sign.
- Negative numbers will be printed in red italic characters with a minus sign.
- Zeros will be printed in green bold, font size increased by 25%, with the text "ZERO".

Here is the result, using a repeating variable / field with that format, or as one column in a Table object (with variable row height to hold the taller ZERO value):

-16 238
-5 526
-31 880
-21 940
+4 137
-100
ZERO
+27 512
-9 330
+21 250
+707
+28 936
-30 953
-24 692
+24 109
-24 352

Note that if the number format is too “small” to hold the number, 4D (and SuperReport Pro) will display it as “<<<<<<<<<<<<<<<<<<”, which will interfere with the opening tag character “<” if the column is attributed (multistyled).

In the example above (using “## ###” as a number format), this will be the case for all numbers exceeding 99,999.

Make sure that the format used will not cause the number to overflow, lest unexpected results might ensue.

Commands by Theme

Using the Command Reference

Each SuperReport Pro command is described in detail in this section.

Each description contains the following elements:

Name of the command

Parameters

Result or Error

SR_ConvertReportTo...ML

(blob:O; XML:T; name:T; callback:T) → error:L

Descriptions of the parameters

Parameter	Type	Description
→ blob	blob	Blob to convert to text.
← XML	text	XML output from the **blob**.
→ name		SRP_Object_Name property.
→ callback		SRP_DataSource_Callback property.

A description of this command

Convert a SuperReport Pro report template from blob to text ir...

Something you should be aware of

Blob can be a SuperReport Pro v2.x document or SuperReport Pr... v3.x XML in a blob (UTF8 or UTF16LE).

Note: ***SR_ConvertReportToXML*** is used internally when needed – you don't have to convert all old reports to use them. If the report is already converted, it is immediately returned without conversion.

Example

One or more examples showing how to use the command

Convert a report that has been saved in a blob field into XML:

```
C_TEXT($reportXML)
$error:=SR_ConvertReportToXML([Reports]ReportBlob;$reportXML;"MyReport";"CallbackMethod")
```

Name of the command

This is what tells SuperReport Pro what you want to do.

The command name must always be entered exactly as shown.

Parameters

Every command requires at least one parameter. Most require the first parameter to be the Area / Report reference.

Result / Error Codes

Functions return a result after they have been called.

Unless otherwise specified in the parameter description table, the result codes are long integers, 0 meaning "no error". Constants are available for these error codes.

Parameter Descriptions

Each command has its own set of parameters, and they are each described in the parameter descriptions table. The tables comprise three columns:Parameter, Type, and Description.

Parameter: The name of the parameter, as shown in the Parameter list. Each is preceded by one of two arrows which indicate whether it is a value that you pass to the command or one that the command returns to you:

→ A value that you pass to the command

← A value that is returned by the command

↔ A value that you pass to the command and / or that can be modified by the command

Type: The type of the parameter. Note that if your database is running in non-Unicode mode, text objects are limited to 32k characters.

Description: Information about the parameter.

Note: when calling a plugin command, all omitted parameters are initialized to the NULL of the respective types (0, 0.0, "", !00:00:00!, …).

Command Description

An explanation of what the command does and how to use it.

Examples

One or more examples demonstrating how the command might be used.

See also Commands and Functions.

Command Themes

The commands are organised into six themes:

Access**:** Commands that affect the entire SuperReport Pro area or the plugin itself

Getters**:** Commands for getting information about certain aspects of the report

Setters**:** Commands for setting properties of objects on the report

Objects**:** Commands for managing the various objects on the report

Printing**:** Commands to manage the printing of SuperReport Pro reports

Miscellaneous: Some additional tools and utilities.

For each theme there is a set of properties that can be used with that theme's commands.

You will find a complete list of properties in the Properties by Theme section.

Access

The commands in this theme affect the overall SuperReport Pro plugin, report or area.

%SuperReport

The actual plug-in area object, which is placed on the 4D form in which you wish to display the SuperReport Pro area.

In addition, you use the ***%SuperReport*** routine when using the 4D **Open external window** function, which provides the ability to display the SuperReport Pro editor in a standard 4D external window. For example:

```
$areaRepRef:=Open external window(50;50;Screen width-50;Screen height-50;8;\
    "New Report";"%SuperReport")
```

SR_ConvertReportToXML

(blob:O; XML:T; name:T; callback:T) → error:L

Parameter	Type	Description
→ blob	blob	Blob to convert to text.
← XML	text	XML output from the **blob**.
→ name	text	SRP_Object_Name property.
→ callback	text	SRP_DataSource_Callback property.

Convert a SuperReport Pro report template from blob to text in XML format.

Blob can be a SuperReport Pro v2.x document or SuperReport Pro v3.x XML in a blob (UTF8 or UTF16LE).

Note: ***SR_ConvertReportToXML*** is used internally when needed – you don't have to convert all old reports to use them. If the report is already converted, it is immediately returned without conversion.

Example

Convert a report that has been saved in a blob field into XML:

```
C_TEXT($reportXML)
$error:=SR_ConvertReportToXML([Reports]ReportBlob;$reportXML;"MyReport";"CallbackMethod")
```

SR_DeleteReport

(areaReportRef:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.

Delete an offscreen report created with SR_NewReport or SR_NewReportBLOB. You should be sure to always delete offscreen reports after you've finished with them.

Example

```
$error:=SR_DeleteReport ($areaRepRef)
```

SR_LoadReport

(areaReportRef:L; src:T; options:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ src	text	Report XML description or path to the report file.
→ options	longint	**src**="", **options**=0: empty XML, empty report – does not contain any section, only contains 2 objects: data source and default style **src**="", **options**=1: empty path, ask for file **src**="<?XML@", **options** is irrelevant: XML is implicit otherwise **src** is a path to load the report from.

Open a saved report and display it in the report area.

Example

This example opens a report that has been saved in a text field:

```
If (Length([Reports]ReportText)>0)
   $error:=SR_LoadReport ($areaRepRef;[Reports]ReportText;0)
End if
```

Compatibility note: this will only work for already converted / new reports.

Old 2.x blob reports must use SR_ConvertReportToXML in Unicode mode only, or the old v2.x API command ***SR Set Area*** (see SuperReport Pro 2.9 manual).

SR_NewReport

(areaReportRef:L;src:T;options:L) → error:L

Parameter	Type	Description
← areaReportRef	longint	Area / Report reference.
→ src	text	Report XML description or path to the report file.
→ options	longint	**src**="", **options**=0: empty XML, empty report – does not contain any section, only contains 2 objects: data source and default style **src**="", **options**=1: empty path, ask for file **src**="<?XML@", **options** is irrelevant: XML is implicit otherwise **src** is a path to load the report from.

Create a new offscreen report, optionally loading a report template.

SR_NewReport expects a text in **src** – the report XML description or a path to the report file. A report loaded from a file can be in old v2.x blob format.

Example 1 – Create a new offscreen report, load a report from a file selected by the user

```
C_LONGINT($error;$areaRepRef)
$error:=SR_NewReport ($areaRepRef;"";1)  // empty source path, ask for a file
```

Example 2 – Create a new offscreen report, load a report from a blob field

```
C_LONGINT($error;$areaRepRef)
C_TEXT($areaXML)
$areaXML:=BLOB to text([Reports]ReportData;UTF8 text without length)
$error:=SR_NewReport ($areaRepRef; $areaXML;0)  // creates empty report if the XML text is empty
```

SR_NewReportBLOB

(areaReportRef:L; src:O) → error:L

Parameter	Type	Description
← areaReportRef	longint	Area / Report reference.
→ src	blob	Report template (blob) to load.

Create a new offscreen report, optionally loading a report template from a blob.

SR_NewReportBLOB expects a blob, which can be in old v2.x blob format.

SR_Register

(registrationCode:T; options:L; email:T) ➞ result

Parameter	Type	Description
➞ registrationCode	text	Pass the registration key to register your copy of SuperReport Pro. The key is either linked to the 4D or 4D Server serial number (individual licenses), to the machine ID (merged licenses), to the name of the company/developer (unlimited annual licenses) or to the product (master key for Online registration).
➞ options	longint	An optional longint combining up to 4 bits.
➞ email	text	Online registration option: developer email to notify when a license is issued or resent.
⬅ result	longint	0 or error code.

SR_Register is used to register the SuperReport Pro plugin for standalone or server use.

Please see the License Types section for detailed information about the licensing options available for SuperReport Pro.

Multiple calls to ***SR_Register*** are allowed. The plugin will be activated if at least one valid key is used, and all subsequent calls to ***SR_Register*** will return 0, unless the force check bit is set to true in the **options** parameter.

registrationCode — You must call ***SR_Register*** with a valid registration key, otherwise SuperReport Pro will operate in demonstration mode – it will cease to function after 20 minutes. In case a master key is used the plugin will attempt a connection to e-Node's license server for Online registration.

options — Optional. This parameter combines up to 4 bits as described below. The default mode (**registrationCode** being a passed as the only parameter) is silent: no force check, no confirmation, no alert, no email.

Bit number	Description
0	Force check: if this bit is is on (true), **registrationCode** is tested regardless of current registration state. If the plugin was not previously registered and the result is 0, it is registered the same way as if the bit was off (or the whole **options** parameter omitted) If the plugin was previously successfully registered, a registration error will be returned in **result** in case **registrationCode** is invalid, but the plugin will remain registered
1	Online registration option: confirm connection "Is it OK to connect to e-Node's license server to register SuperReport Pro?"
2	Online registration option: display alert if registration error
3	Online registration option: display alert if registered

email — Optional. The developer email address where to send Online registration information.

result — 0 or error code:

Result code	Description
0	OK
1	Beta license has expired
2	Invalid license
3	The license has expired
4	The OEM license has expired
5	The maximum number of users has been exceeded
6	The license is for a different environment (e.g. the licence is for a single-user version, but you are using it with 4D Server)
7	The license is linked to a different 4D license
8	Invalid merged license
9	Only serial/ID licenses are allowed in text license files (includes Register button and Online registration)
10	Unauthorized master key (Online registration)
11	Can't connect to e-Node's license server to perform Online registration
12	No Online registration license available for this master key (unknown or all used)

When ***SR_Register*** is called with an empty string, the license dialog will be displayed if SuperReport Pro is not registered and the dialog was not yet displayed. This allows you to show the registration dialog to your users without effectively calling a SuperReport Pro command or displaying a SuperReport Pro area.

Note: alternately to ***SR_Register***, you can place a plain text file into your 4D Licenses folder or use the Demo mode dialog "Register" button. This is only valid for non-unlimited licenses.

Basic example

```
C_LONGINT ($result)
$result:=SR_Register ("YourRegistrationKey")
Case of
   :($result=2)
      ALERT ("The SuperReport Pro licence is invalid.")
   :($result=3)
      ALERT ("The SuperReport Pro licence has expired.")
   etc.
End case
```

Example with multiple calls

```
C_LONGINT ($result)  //ignored in this case
$result:=SR_Register ("Registration key one")
$result:=SR_Register ("Registration key two")
$result:=SR_Register ("Registration key three")
```

etc.

```
If ($result#0)  //registration failed on all keys
   ALERT ("SuperReport Pro could not be registered.")
End if
```

Force check example

In this example we assume that only "Registration key two" is valid, but you want to check the other keys status.

```
C_LONGINT ($result)
$result:=SR_Register ("Registration key one";1)  //invalid, will return an error, the plugin isn't registered
$result:=SR_Register ("Registration key two";1)  //valid, will return 0, the plugin is registered
$result:=SR_Register ("Registration key three";1)  //invalid, will return an error, the plugin is still registered
```

Online registration examples

Confirm connection, alert if successful, alert if failed, send email notification to developer@4dchampions.com:

```
C_LONGINT ($result)
$result:=SR_Register ("Master key";0 ?+1 ?+2 ?+3;"developer@4dchampions.com")
```

Silent connection, alert if successful, alert if failed, no email notification:

```
C_LONGINT ($result)
$result:=SR_Register ("Master key";0 ?+2 ?+3)
```

SR_SaveReport

(areaReportRef:L; dstpath / XML:T; options:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ dstpath / ← XML	text	The full pathname to a new (will be created) or existing document, or a valid text variable to receive the XML. If the parameter is a null string, the standard **Save File** dialog will be displayed.
→ options	longint	0: **dstpath / XML** will be filled with XML 1: **dstpath / XML** is a path, ask if empty.

Save the contents of the defined report to a file or as XML in a variable.

Using this command with a document pathname does the same as getting SRP_Object_XML from the report / area and saving it to the disk.

This is similar to selecting the **Save to Disk** menu item from the **File** menu in the SuperReport Pro report area, except that using the menu modifies the document name (SRP_Report_Document) and changes the window title if using an external window.

Example – Save a report to a field

```
$areaXML:=""
$error:=SR_SaveReport ($areaRepRef;$areaXML;0)  // save the report data to a variable
If ($error=0)  // if the command completed successfully
   [Customer]customer_report:=$areaXML
End if
```

Getters

Use these commands to get information about a report or object.

The properties that can be used with these commands can be found in the Properties by Theme section.

Use **areaReportRef** = 0 to access a global plugin property (such as SRP_Area_Path or SRP_Area_Version).

- **areaReportRef** = 0 means “access workstation global settings”.
- **areaReportRef** # 0 means “access this area's settings”.

areaReportRef is the SuperReport Pro Area / Report reference.

objNum is 0 = selected objects, 1 = the report itself, any other number = the Object number.

Note: not every object supports every property.

SR_GetLongProperty

(areaReportRef:L; objNum:L; property:T) → value:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself any other value: the object number.
→ property	text	Property constant.
← value	longint	Property value.

Get a longint property **value** of a report or object.

Example 1

Find out which tool in SuperReport Pro's toolbar the user has selected:

```
$toolID:=SR_GetLongProperty ($areaRepRef;1;SRP_Area_Tool)
```

Example 2

Get the number of footer rows for a Table object (Object number $tableobjNum):

```
$numFooterRows:=SR_GetLongProperty ($areaRepRef;$tableobjNum;SRP_Table_NumFooters)
```

SR_GetObjectXML

(areaReportRef:L; objNum:L; XML:T) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself any other value: the object number.
← XML	text	The XML of the specified object.

Get the **XML** value of a report or object and store it into a text variable.

This command is equivalent to the following:

SR_GetTextProperty($areaReportRef;$objNum;SRP_Object_XML)

SR_GetProperties

(areaReportRef:L; objNum:L; properties:AT; values:AT; names:AT) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	0 = selected objects 1 = the report itself any other value: the object number.
← properties	text array	Array of property constants.
← values	text array	Array of property values (converted to text as needed).
← names	text array	(Optional) array of property names.

Get all (relevant) Properties of a report, object or object selection into parallel arrays.

Note: because a text array is used for values, pictures and blobs are unusable and real numbers use a decimal point regardless of the current system settings.

Also, when operating on multiple objects (selection: **objNum**=0), only properties common to all objects are returned and when there are different values for a property, "<multiple values>" is returned.

Example

Get the **properties** of all objects on the report:

```
ARRAY TEXT($propertyIDs;0)
ARRAY TEXT($propertyValues;0)
ARRAY TEXT($propertyNames;0)
$error:=SR_GetProperties ($areaRepRef;1;$propertyIDs;$propertyValues;$propertyNames)
```

See also SR_SetProperties and its example.

SR_GetPtrProperty

(areaReportRef:L; objNum:L;property:T; value;P) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself any other value: the object number.
→ property	text	Property constant.
← value	pointer	Pointer to an object (4D variable or field) containing the property value.

Get a **property value** of a report or object using a pointer.

Example 1 – Using a blob

```
C_BLOB($blob)
$error:=SR_GetPtrProperty ($areaRepRef;1;SRP_Report_UserBLOB;->$blob)
```

Example 2 – Using a pointer

```
C_POINTER($ptr)  //pointer to the area
$error:=SR_GetPtrProperty ($areaRepRef;1; SRP_Area_Self;->$ptr)  //pointer to the $ptr pointer
```

SR_GetRealProperty

(areaReportRef: L;objNum:L; property:T) →value:R

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself any other value: the object number.
→ property	text	Property constant.
← value	real	Property value.

Get a real **property value** of a report or object.

Example

Get the current step for moving / resizing objects with arrow keys in points for the report:

```
C_REAL($step)
$step :=SR_GetRealProperty($areaRepRef;1; SRP_Area_Moving)
```

SR_GetTextProperty

(areaReportRef:L; objNum:L; property:T) →value:T

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself any other value: the object number.
→ property	text	Property constant.
← value	text	Property value, converted to text as needed.

Get a text **property value** of a report or object.

Note: because a text conversion is used for numeric property values, pictures and blobs are unusable and real numbers use a decimal point regardless of the current system settings.

Example

Get the name of the area object (the area's variable name used on the 4D form):

```
C_TEXT($areaVariableName)
$areaVariableName :=SR_GetTextProperty($areaRepRef;1;SRP_Area_Name)
```

Setters

Use these commands to set information about a report or object.

For a list of properties, see the Properties section.

Note: not every object supports every property.

SR_SetLongProperty

(areaReportRef:L; objNum:L;property:T; value:L)

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	0 = selected objects 1 = the report itself any other value: the object number.
→ property	text	Property constant.
→ value	longint	Property value.

Set a longint **property value** of a report or object selection.

Example 1

Hide the SuperReport Pro menu bar:

SR_SetLongProperty ($areaRepRef;1;SRP_Report_ShowMenubar;0)

Example 2

Use PDF for preview, don't embed fonts on Windows:

SR_SetLongProperty (0; 0; SRP_Area_PreviewFlags; SRP_Print_WinPDFPreview | SRP_Print_WinPDFNoFonts)

Note: preview always means “Open the file after creation” – see Preview.

Example 3

Create an OXPS preview on Windows 8 (will open after creation) unless SRP_Print_WinPDFPreview has been set:

SR_SetTextProperty (0; 0; SRP_Area_WinPreviewName; "MyGreatestAppPreview.oxps")

Note: Windows 7 does not support OXPS!

SR_SetProperties

(areaReportRef:L; objNum:L; properties:AT; values:AT) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	0 = selected objects 1 = the report itself any other value: the object number.
→ properties	text array	Array of property constants.
→ values	text array	Array of property values (as text).

Set Properties of a report, object or object selection from parallel arrays. You can set one or multiple **properties**.

Note: because a text array is used for **values**, pictures and blobs are unusable and real numbers use a decimal point regardless of the current system settings.

Example

Let's get report boolean information such as grid, margins, menu bar, borders and ruler show / hide, displaying a custom dialog or palette so that the user can modify these **properties** using checkboxes then apply the changes to the report.

```
 //Dialog form method
ARRAY TEXT($propertyIDs;0)
ARRAY TEXT($propertyValues;0)
C_LONGINT($i;$error)
C_LONGINT(cbxGrid;cbxMargins;cbxMenu;cbxBorders;cbxRuler)  //checkboxes for the dialog
Case of
   : (Form event=On Load)
      $error:=SR_GetProperties ($areaRepRef;1;$propertyIDs;$propertyValues)  //get the values
      For ($i;1;Size of array($propertyIDs))
         Case of
            : ($propertyIDs{$i}=SRP_Report_ShowGrid)
               cbxGrid:=Num($propertyValues{$i})  //0 or 1
            : ($propertyIDs{$i}=SRP_Report_ShowMargins)
               cbxMargins:=Num($propertyValues{$i})
            : ($propertyIDs{$i}=SRP_Report_ShowMenubar)
               cbxMenu:=Num($propertyValues{$i})
            : ($propertyIDs{$i}=SRP_Report_ShowObjBorders)
               cbxBorders:=Num($propertyValues{$i})
            : ($propertyIDs{$i}=SRP_Report_ShowRuler)
               cbxRuler:=Num($propertyValues{$i})
         End case
      End for
```

```
: (Form event=On Validate)
   ARRAY TEXT($propertyIDs;0)  // reset the arrays
   ARRAY TEXT($propertyValues;0)
   APPEND TO ARRAY($propertyIDs;SRP_Report_ShowGrid)
   APPEND TO ARRAY($propertyValues;String(cbxGrid))  // "0" or "1"
   APPEND TO ARRAY($propertyIDs;SRP_Report_ShowMargins)
   APPEND TO ARRAY($propertyValues;String(cbxMargins))
   APPEND TO ARRAY($propertyIDs;SRP_Report_ShowMenubar)
   APPEND TO ARRAY($propertyValues;String(cbxMenu))
   APPEND TO ARRAY($propertyIDs;SRP_Report_ShowObjBorders)
   APPEND TO ARRAY($propertyValues;String(cbxBorders))
   APPEND TO ARRAY($propertyIDs;SRP_Report_ShowRuler)
   APPEND TO ARRAY($propertyValues;String(cbxRuler))
   $error:=SR_SetProperties ($areaRepRef;1;$propertyIDs;$propertyValues)  // set the values
End case
```

See also SR_GetProperties and its example.

SR_SetPtrProperty

(areaReportRef:L; objNum:L; property:T; value:P) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	0 = selected objects 1 = the report itself.
→ property	text	Property constant.
→ value	pointer	Pointer to an object (4D variable or field) containing the property value.

Set a **property value** of a report or object selection using a pointer.

Example 1

Clear the page setup settings for a report prior to saving it:

```
C_BLOB($blob)
$error:=SR_SetPtrProperty ($areaRepRef;1;SRP_Report_PageFormat;->$blob)  // MacOS
$error:=SR_SetPtrProperty ($areaRepRef;1;SRP_Report_PrintSettings;->$blob)
$error:=SR_SetPtrProperty ($areaRepRef;1;SRP_Report_DevMode;->$blob)  // Windows
$error:=SR_SetPtrProperty ($areaRepRef;1;SRP_Report_DeviceNames;->$blob)
```

Example 2

Get, then restore the print settings:

```
C_LONGINT($error)
C_BLOB($blob)
$error:=SR_GetPtrProperty ($areaRepRef;1;SRP_Report_PrintSettings;->$blob)
   // do something
$error:=SR_SetPtrProperty ($areaRepRef;1;SRP_Report_PrintSettings;->$blob)
```

SR_SetRealProperty

(areaReportRef:L; objNum:L; property:T; value:R)

Parameter	Type	Description
➞ areaReportRef	longint	Area / Report reference.
➞ objNum	longint	0 = selected objects 1 = the report itself.
➞ property	text	Property constant.
➞ value	real	Property value.

Set a real **property value** of a report or object selection.

Example

Zoom the report in the editor to 150 %

```
SR_SetRealProperty ($areaRepRef;1;SRP_Report_Zoom;1.5)
```

SR_SetTextProperty

(areaReportRef:L; objNum:L; property:T; value:T)

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	0 = selected objects 1 = the report itself.
→ property	text	Property constant.
→ value	text	Property value.

Set a text **property value** of a report or object selection.

Note: because a text conversion is used for numeric property values, pictures and blobs are unusable and real numbers use a decimal point regardless of the current system settings.

Example

Set a text object font to Arial:

```
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Style_FontName;"Arial")
```

Objects

SR_ChangeObjectParent

(areaReportRef:L; objNum:L; newParent:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	The object number (value > 1).
→ newParent	longint	Reference number of the new parent object.

Move an object from one section / group into another section / group.

Note: this command does not support **objNum**=1 for the report itself. It will return the error code -5 if used.

SuperReport Pro version 3 uses a real object hierarchy – the objects are contained in the section / group.

When you programmatically change the position, it will not move to a different parent.

An object's position is relative to the parent, so when you change an object's parent, the relative position will be the same.

Example

In this example we will create a new group as an object, in which we'll include the current object selection.

```
   // Group selected objects
C_LONGINT($1)  // report
C_LONGINT($error;$group;$i;$parent;$areaRepRef;$lock)
C_REAL($top;$left;$bottom;$right;$lineWidth)
C_REAL($gtop;$gleft;$gbottom;$gright)
C_TEXT($kind)
$areaRepRef:=$1
ARRAY LONGINT($objectNums;0)
$error:=SR_GetObjects ($areaRepRef;1;SRP_ReportSelectedObjects;$objectNums)
   // Eliminate unsupported objects
For ($i;Size of array($objectNums);1;-1)
   $kind:=SR_GetTextProperty ($areaRepRef;$objectNums{$i};SRP_Object_Kind)
   If (($kind=SRP_ObjectKind_Report)\
      | ($kind=SRP_ObjectKind_Style)\
      | ($kind=SRP_ObjectKind_Guide)\
      | ($kind=SRP_ObjectKind_Section)\
      | ($kind=SRP_ObjectKind_Header)\
      | ($kind=SRP_ObjectKind_Column)\
      | ($kind=SRP_ObjectKind_Footer))
```

```
      DELETE FROM ARRAY($objectNums;$i;1)
   End if
End for

If (Size of array($objectNums)>1)
 // Get bounding box
   For ($i;1;Size of array($objectNums))
      $top:=SR_GetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_PosTop)
      $left:=SR_GetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_PosLeft)
      $bottom:=SR_GetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_PosBottom)
      $right:=SR_GetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_PosRight)
      $lineWidth:=0
      $error:=SR_GetPtrProperty ($areaRepRef;$objectNums{$i};SRP_Line_Thickness;->$lineWidth)
      If ($i=1)
         $gtop:=$top-$lineWidth
         $gleft:=$left-$lineWidth
         $gright:=$right+$lineWidth
         $gbottom:=$bottom+$lineWidth
      Else
         If ($gtop>($top-$lineWidth))
            $gtop:=$top-$lineWidth
         End if
         If ($gleft>($left-$lineWidth))
            $gleft:=$left-$lineWidth
         End if
         If ($gbottom<($bottom+$lineWidth))
            $gbottom:=$bottom+$lineWidth
         End if
         If ($gright<($right+$lineWidth))
            $gright:=$right+$lineWidth
         End if
      End if
   End for
   $error:=SR_Area_SaveUndo ($areaRepRef;219)  // start undo block for "Group"
    // Create the group object
   $error:=SR_GetParent ($areaRepRef;$objectNums{1};$parent)
   $error:=SR_NewObject ($areaRepRef;$group;SRP_Group;$parent)
   SR_SetRealProperty ($areaRepRef;$group;SRP_Object_PosTop;$gtop)
   SR_SetRealProperty ($areaRepRef;$group;SRP_Object_PosLeft;$gleft)
```

```
   SR_SetRealProperty ($areaRepRef;$group;SRP_Object_PosBottom;$gbottom)
   SR_SetRealProperty ($areaRepRef;$group;SRP_Object_PosRight;$gright)
   SR_SetLongProperty ($areaRepRef;$group;SRP_Object_VariableSizeV;1)
   SR_SetLongProperty ($areaRepRef;$group;SRP_Object_Selected;1)
    //Move objects into the group
   $gtop:=-$gtop
   $gleft:=-$gleft
   $lock:=2  //locked mode
   For ($i;1;Size of array($objectNums))
      SR_SetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_RelMoveH;$gleft)
      SR_SetRealProperty ($areaRepRef;$objectNums{$i};SRP_Object_RelMoveV;$gtop)
      SR_SetLongProperty ($areaRepRef;$objectNums{$i};SRP_Object_Selected;0)
      $error:=SR_ChangeObjectParent ($areaRepRef;$objectNums{$i};$group)
      SR_SetLongProperty ($areaRepRef;$objectNums{$i};SRP_Object_Locked;$lock)
   End for
   $error:=SR_Area_SaveUndo ($areaRepRef;0)  //end undo block
End if
```

Note that the above action can also be performed directly with an **Object > Group** menu call using the old v2 API ***SR Do Command***:

```
$error:=SR Do Command ($1;219;1)  //Object > Group menu
```

SR_DeleteObject

(areaReportRef:L; objNum:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	The object number (value > 1).

Delete the specified object.

Example

```
$error:=SR_DeleteObject($areaRepRef;3)   //delete object number 3 from the report
```

SR_FindObjectByID

(areaReportRef:L; objID:T; objNum:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objID	text	Object ID.
← objNum	longint	Object number.

Return (in **objNum**) the Object number of the first object in a report having its Object ID equal to **objID**.

The comparison is strictly equal: it is case and diacritical sensitive.

Example

```
C_LONGINT($objNum)
$error:=SR_FindObjectByID($areaRepRef;"Header";$objNum)
```

SR_GetObjects

(areaReportRef:L; objNum:L; mode:T; objNumArray:AL) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	Refers to report, section, group or a table object. 1 = the report itself any other value: the object number.
→ mode	text	SRP_ReportAllObjects SRP_ReportSelectedObjects SRP_ReportEditorStyles SRP_ReportDataSource SRP_ReportGuides SRP_ReportStyleSet SRP_ReportSections SRP_GroupObjects SRP_TableHeaderRowMask* SRP_TableColumns SRP_TableFooterRowMask*
← objNumArray	longint array	Array containing all Object numbers.

*Use that mask and add the header row number (starting at 1):
String ($row; SRP_TableHeaderRowMask). So for example the third header line is "H003".

See for example Creating Reports Procedurally.

Get the Object numbers of the objects included in a report, section, group or Table object.

Example 1

Get the table (Object number $tableobjNum) columns, and then set the style ID for the column header of the first column to 2:

```
ARRAY LONGINT($objectNums;0)
$error:=SR_GetObjects ($areaRepRef;$tableobjNum;SRP_TableColumns;$objectNums)  //get the columns
SR_SetLongProperty ($areaRepRef;$objectNums{1};SRP_Object_StyleID;2)
```

Example 2

Get cells in the footer row 1:

```
ARRAY LONGINT($objectNums;0)
$error:=SR_GetObjects($areaRepRef;$tableobjNum;String(1;SRP_TableFooterRowMask);$objectNums)  //"F001"
```

SR_GetObjectsByPropertyValue

(areaReportRef:L; property:T; value:T; objNumArray:AL; comparisonMode:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ property	text	Property constant.
→ value	text	Value to look for (converted to text as needed).
← objNumArray	longint array	Array containing the Object numbers of the objects matching the property value.
→ comparisonMode	longint	Comparison mode (optional): Mode: 0 – Smooth (ignore case & diacritics) – default value 1 – Case insensitive (ignore case) 2 – Accentless (ignore diacritics) 3 – Strictly equal

Get the Object numbers of the objects matching the **value** for the specified **property**.

Note: the **Property** can be any property accepting a type that can be converted to text. Because a text conversion is used for numeric property values, pictures and blobs are unusable and real numbers use a decimal point regardless of the current system settings.

Example

Get all instances of a given 4D variable on the report:

```
ARRAY LONGINT($objectNums;0)  //will contain the Object numbers
$error:=SR_GetObjectsByProperty ($areaRepRef; SRP_Variable_Source; "vVarName"; $objectNums)
```

Note that the above equals to the following:

```
$property:=SRP_Variable_Source
$value:="vVarName"  // name of the 4D variable we're looking for
ARRAY LONGINT($allObjectNums;0)
SR_GetObjects ($areaRepRef; SRP_ReportAllObjects; $allObjectNums)  // get all Object numbers
ARRAY LONGINT($objectNums;0)
C_TEXT($v)
For ($i;1;Size of array ($allObjectNums))
   If (SR_GetPtrProperty ($areaRepRef; $allObjectNums{$i};$property;->$v)=0)  // object property value
      If ($v=$value)  // requested value
         APPEND TO ARRAY ($objectNums;$allObjectNums{$i})
      End if
   End if
End for
```

SR_GetParent

(areaReportRef:L; objNum:L; parentObjNum:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	1 = the report itself (will return 0) any other value: the object number.
← parentObjNum	longint	The object's parent object number.

Return (in **parentObjNum**) the Object number of the object's parent.

Example

```
C_LONGINT($parentobjNum)
$error:=SR_GetParent ($areaRepRef;3;$parentobjNum)  // get the parent object number to the object number 3.
```

SR_ModifyTable

(areaReportRef:L; objNum:L; action:L; from:L; numColumns:L; to:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ objNum	longint	Table object to modify.
→ action	longint	1 = insert **numColumns** (> 0) columns before column **from** (> 0, <= number of columns in the table) 2 = move **numColumns** (> 0) columns at position **from** (> 0, <= number of columns in the table) after column **to** 3 = delete **numColumns** (> 0) columns at position **from**.
→ from	longint	
→ numColumns	longint	
→ to	longint	

Insert, move or delete column(s) of the specified Table.

Example

```
$error:=SR_ModifyTable ($areaRepRef;6;2;3;2;5)  // move the table (object number 6) columns 3 and 4 after column 5.
```

SR_NewObject

(areaReportRef:L; objNum:L; type:T; parentObjNum:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
← objNum	longint	The returned new object number.
→ type	text	SRP_Group SRP_Line SRP_Oval SRP_Rectangle SRP_Picture SRP_Text SRP_Variable SRP_Field SRP_Table SRP_Header SRP_BreakHeader SRP_Body SRP_BreakFooter SRP_Footer SRP_Watermark SRP_Scrap SRP_HorizontalGuide SRP_VerticalGuide SRP_DataSource SRP_Style
→ parentObjNum	longint	Parent section or group object number, must be provided for “normal” objects (not guide, style, section).

Create a new object within the given parent object.

Example

Create a new header section, get its Object number, then assign a name to it and set its height to 40 points:

```
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Header;0)  // 0 is ignored (section)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"Header")
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Section_Height;40)
```

SR_NewObjectFromXML

(areaReportRef:L; objNum:L; XML:T; parentObjNum:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
← objNum	longint	The returned new object number.
→ XML	text	XML defining the object.
→ parentObjNum	longint	Parent section or group object number, must be provided for “normal” objects (not guide, style, section).

Create a new object from an **XML** snippet within the given parent.

You can create an object's **XML** programmatically and then use it to create a new object. It can be any kind of object except the report itself.

The result is the same as creating an object using ***SR_NewObject*** and then setting all the needed properties.

Example

The creation of styles in the Creating Reports Procedurally example could be rewritten this way:

```
$XML:="<Style id=1 name=Headers font=Helvetica size=8 qdStyle=1 wrap=1 lineSpacing=1/>"
$error:=SR_NewObjectFromXML ($areaRepRef;$objNum;$XML;0)  // create style 1
$XML:="<Style id=2 name=Headers font=Helvetica size=8 wrap=1 lineSpacing=1/>"
$error:=SR_NewObjectFromXML ($areaRepRef;$objNum;$XML;0)  // create style 2
```

Note: the above XML is not strictly orthodox – attributes must normally use the syntax **attribute="value"** or **attribute='value'**. SuperReport Pro supports omission of quotes (as above) but multi-word values must be quoted.

Printing

SR_AbortPrinting

Cancels current printing (no parameters, no result value).

This command can only be called from a script (Report or Object) or a 4D project method called from a script.

Does nothing if no printing is in progress.

Example

See the example for Execution Cycle scripts.

SR_CloseSession

(session:L) → error:L

Parameter	Type	Description
➞ session	longint	Print session ID.

Close a printing session.

Example

See the example for SR_OpenSession.

SR_Export

(src:T;options:L; fieldDelimiter:L; dstPath:T; recordDelimiter:L) → error:L

Parameter	Type	Description
→ src	text	Report XML description or path to the report file.
→ options	longint	**src**="", **options** bit 0=0: empty XML **src**="", **options** bit 0=1: empty path, ask for file **src**="<?XML@", **options** bit 0 is irrelevant: XML is implicit otherwise **src** is a path to load the report from. For other bits, use SuperReport Pro Export Flags: - set exactly one of SRP_Export_Text, SRP_Export_XML, or SRP_Export_HTML - for text export, SRP_Export_CSVFormat can be used (export 1,2 as "1,2") to escape special characters - set at least one of SRP_Export_Body, SRP_Export_Breaks, SRP_Export_Total, SRP_Export_Headers, SRP_Export_Watermark. - set SRP_Export_StaticText to export static texts - set SRP_Export_Sorted to export in print order, not visual order (position on page).
→ fieldDelimiter	longint	ASCII code of the field delimiter.
→ dstPath	text	A valid document pathname. If it is a null string, the standard **Save File** dialog will be displayed.
→ recordDelimiter	longint	A long integer which can contain 2 characters where the first character is put in low-order word. The default is "\r\n" which is equivalent to Carriage Return | (Line Feed << 16). The record delimiter is always CR+LF in XML and HTML modes.
← error	longint	See the error numbers listed for SR_Print.

Export a SuperReport Pro report without requiring a SuperReport Pro area – the output written to the specified document is a text interpretation of the report.

Note: the Header and Footer sections are only used once as the report is generated since the concept of pages don't exist when exporting the report to a document.

Example 1 – Export the Body section of a report to an XML file

```
$error:=SR_Export ($areaRepRef;SRP_Export_XML | SRP_Export_Body;9;\
    "Macintosh Disk:ProjectFolder:Report:MyReport.XML";13)
```

Example 2 – Export the Body section of a report to a CSV text file

```
$error:=SR_Export ($areaRepRef; SRP_Export_Text | SRP_Export_CSVFormat | SRP_Export_Body;9;\
    "Macintosh Disk:ProjectFolder:Report:MyReport.csv";13)
```

SR_ExportBLOB

(src:O;options:L; fieldDelimiter:L; dstPath:T; recordDelimiter:L) → error:L

This is the same as SR_Export, except that the source is a blob and **options** bit 0 is not used.

SR_ExportBLOBIntoBLOB

(src:O;options:L; fieldDelimiter:L; dstBlob:O; recordDelimiter:L) → error:L

This is the same as ***SR_ExportBLOB***, except that both the source and the destination are blobs.

SR_ExportIntoBLOB

(src:T;options:L; fieldDelimiter:L; dstBlob:O; recordDelimiter:L) → error:L

This is the same as ***SR_Export***, except that the destination is a blob instead of a file.

Example

Export a report into a blob as HTML.

```
C_BLOB($blob)
SET BLOB SIZE($blob;0)
$options:=SRP_Export_HTML
$error:=SR_ExportIntoBLOB ([Report]ReportData;$options;9;$blob;13)
```

SR_OpenSession

(session:L; dstFlags:L; dstPath:T; XML:T; jobName:T; printer:T) → error:L

Parameter	Type	Description
← session	longint	Print session ID.
→ dstFlags	longint	Destination flags – see Print Flags for the options.
→ dstPath	text	Document path when printing to PDF or file.
→ XML	text	A template used just for Page setup / Job setup.
→ jobName	text	Printer job name.
→ printer	text	Printer name (optional): - used if non empty - otherwise and if SRP_Print_NoDefaultPrinter is set to true in **dstFlags**, the printer stored in the template is used - otherwise the default printer is used.

Start a new printing **session**. Sessions are a way to print multiple reports as one print job.

Note: preview always means “Open the file after creation” – see Preview.

The SRP_Print_DestinationPDF and SRP_Print_DestinationFile option bits (destination flags in **dstFlags**) require a file path in **dstPath**.

If it is empty, on Windows ***SR_OpenSession*** will behave the same as if SRP_Print_DestinationPrinter was used and nothing will happen on MacOS.

Example

Print reports of your top 20 customers and your top 50 customers as one print job:

```
$dest:=SRP_Print_AskPageSetup
$error:=SR_OpenSession ($session;$dest;"";[Reports]RM_ReportData;"Top customers";"LaserJet")
QUERY([Reports];[RM_Reports]RM_ReportName="Top 20 Customers")
$error:=SR_Print ([Reports]RM_ReportData;0;0;"";$session)
QUERY([Reports];[Reports]RM_ReportName="Top 50 Customers")
$error:=SR_Print ([Reports]RM_ReportData;0;0;"";$session)
$error:=SR_CloseSession ($session)
```

SR_OpenSessionBLOB

(session:L; dstFlags:L; dstPath:T; XML:O; jobName:T; printer:T) → error:L

Parameter	Type	Description
← session	longint	Print session ID.
→ dstFlags	longint	Destination flags – see Print Flags for the options.
→ dstPath	text	Document path when printing to PDF or file.
→ XML	blob	A template used just for Page setup / Job setup.
→ jobName	text	Printer job name.
→ printer	text	Printer name.

Same as ***SR_OpenSession*** but with default PageSetup / JobSetup XML stored into a blob instead of a text variable or field.

SR_Print

(src:T; options:L; dstFlags:L; dstPath:T; session:L; printer:T; count:L) → error:L

Parameter	Type	Description
→ src	text	Report XML description or path to the report file.
→ options	longint	**src**="", **options**=0: empty XML **src**="", **options**=1: empty path, ask for file **src**="<?XML@", **options** is irrelevant: XML is implicit otherwise **src** is a path to load the report from.
→ dstFlags	longint	Destination flags – see Print Flags for the options.
→ dstPath	text	Document path when the destination is a file.
→ session	longint	If not null, it will be used for printing and **dstFlags** / **dstPath** / **printer** will be ignored (and PageSetup / JobSetup in **src** will be ignored).
→ printer	text	Printer name.
↔ count	longint	If not null, the report is not printed; instead, the number of pages is returned into this parameter.
← error	longint	-205: Header / Footer section does not fit on a page. -206: object does not fit into section / group (probably is outside the bounds of a fixed height section / group) -207: object does not fit into fixed-size group (outside the bounds of a group) -210: Header section does not fit on a page -211: section does not fit on a page -212: too many sub-pages (horizontal splitting of a Table object) -213: too many pages (probably requesting a new page infinitely) -214: section does not fit on a new page

Start a new printing session if **session** is zero, produces a report within the new or specified **session**, ends the printing session if **session** is zero.

Note: preview always means “Open the file after creation” – see Preview.

Example

You can use ***SR_Print*** to create a PDF directly.

On MacOS (uses native PDF support):

```
$error:=SR_Print ($areaXML;0;SRP_Print_DestinationPDF;\
    "Macintosh Disk:ProjectFolder:Report:MyReport.pdf";0;"")
```

On Windows (uses built-in PDF generator):

```
$error:=SR_Print ($areaXML;0;SRP_Print_DestinationPDF;\
    "C:\\ProjectFolder\\Printing\\MyReport.pdf";0;"")
```

SR_PrintBLOB

(src:O; options:L; dstFlags:L; dstPath:T; session:L; printer:T; count:L) → error:L

Parameter	Type	Description
→ src	text	Source blob to print.
→ options	longint	Not used.
→ dstFlags	longint	Destination flags – see Print Flags for the options.
→ dstPath	text	Document path when the destination is a file.
→ session	longint	If not null, it will be used for printing and **dstFlags** / **dstPath** / **printer** will be ignored (and PageSetup / JobSetup in **src** will be ignored).
→ printer	text	Printer name.
↔ count	longint	If not null, the report is not printed; instead, the number of pages is returned into this parameter.

Same as ***SR_Print*** but from a blob instead of a text file or XML as text. The **options** parameter is not relevant.

SR_PrintBLOBIntoPICT

(src:O; options:L; dstFlags:L; dstArray:AP; dstFormat:L; printer:T) → error:L

Parameter	Type	Description
→ src	blob	The blob to print.
→ options	longint	Not used.
→ dstFlags	longint	See Print Flags – but the destination is ignored – only the PageSetup bits are usable.
→ dstArray	picture array	The array should be initialised as **ARRAY PICTURE**; If the array is not empty, the pictures will be added.
→ dstFormat	longint	0 – default - vector graphics (SRP_PrintPict_PDFVector - PDF on Mac, SRP_PrintPict_EMFVector – EMF on Windows) 1 – SRP_PrintPict_PNG 2 – SRP_PrintPict_TIFF 3 – SRP_PrintPict_JPEG 4 – SRP_PrintPict_BMP 5 – SRP_PrintPict_GIF 6 – SRP_PrintPict_PDFBitmap (on MacOS) Bitmaps are generated at 72 DPI.
→ printer	text	The printer driver to use.

Print the specified blob into a picture.

SR_PrintIntoPICT

(src:T; options:L; dstFlags:L; dstArray:AP; dstFormat:L; printer:T) → error:L

Parameter	Type	Description
→ src	text	A valid SuperReport Pro Object. This is not the actual SuperReport Pro area but rather the report object as it would be stored in a 4D blob field or variable.
→ options	longint	0 = **src** is XML 1 = **src** is a file name; ask for the file name is src is empty Ignored if **src**="<?XML@": **src** is XML.
→ dstFlags	longint	See Print Flags – but the destination is ignored – only the PageSetup bits are usable.
→ dstArray	picture array	The array should be initialised as **ARRAY PICTURE**. The picture array items will be appended whether the array is empty or not.
→ dstFormat	longint	0 – default - vector graphics (SRP_PrintPict_PDFVector - PDF on Mac, SRP_PrintPict_EMFVector – EMF on Windows) 1 – SRP_PrintPict_PNG 2 – SRP_PrintPict_TIFF 3 – SRP_PrintPict_JPEG 4 – SRP_PrintPict_BMP 5 – SRP_PrintPict_GIF 6 – SRP_PrintPict_PDFBitmap (on Mac) Bitmaps are generated at 72 DPI.
→ printer	text	The printer driver to use.

Print the report into a picture.

SR_PrintSettings

(areaReportRef:L; options) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ options		A combination of: SRP_Print_ValidatePageSetup SRP_Print_DefaultPageSetup SRP_Print_4DPageSetup SRP_Print_SimplePageSetup * SRP_Print_DefaultJobSetup SRP_Print_4DJobSetup SRP_Print_AskPageSetup SRP_Print_AskJobSetup

* Use only the report size, orientation and scaling stored in the report, not platform-native page setup / job setup.

Set various printing options.

Example

```
$error:=SR_PrintSettings ($areaRepRef;SRP_Print_ValidatePageSetup)
```

Miscellaneous

SR_Area_Redo

(areaReportRef:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.

Execute the next redo action from the Undo stack.

Example

```
$error:=SR_Area_Redo($areaRepRef)
```

SR_Area_SaveUndo

(areaReportRef:L; operation:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.
→ operation	longint	-1 = pause -2 = resume 0 = stop anything else: start.

SuperReport Pro contains unlimited undo support. The plugin records all user actions and inserts them into the undo stack.

Changes executed to the report through code must be inserted into the undo stack explicitly with the ***SR_Area_SaveUndo*** command.

The **operation** parameter is used for naming **Undo / Redo** menu items. Internally, menu command IDs are used, e.g. 219 = **Group objects** (see the example for SR_ChangeObjectParent).

Example

```
// Start recording actions
$error:=SR_Area_SaveUndo ($areaRepRef;1)
// Execute any commands
// ...
// Stop recording and save actions to undo stack
$error:=SR_Area_SaveUndo($areaRepRef;0)
```

SR_Area_Undo

(areaReportRef:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.

Execute the next undo action from the Undo stack.

Example

```
$error:=SR_Area_Undo($areaRepRef)
```

SR_ColorPicker

(color:L; actionValue:L) → OK:L

Parameter	Type	Description
↔ color	longint	On input: the default color to preselect On output: the selected color
↔ actionValue	longint	On input: 0 – show platform native color picker 1 – show 4D RGB color picker (supports alpha channel) 2 – don't show a picker On output: **Color** mixed with white background.

Invoke the color picker, optionally setting a default **color**. The returned value is 1 if OK, 0 otherwise (user cancelled).

Example 1 – Use the native color picker

```
$color:=-65536
$SRok:=SR_ColorPicker($color;0)
```

Example 2 – Use the 4D form color picker

```
$color:=0xFFFF0000  // full red
$SRok:=SR_ColorPicker($color;1)
```

Example 3 – Use the 4D form color picker, return the mixed value

```
$color:=0x80FF0000  // 50% transparent red
$mixed:=1
$SRok:=SR_ColorPicker($color;$mixed)
```

Assuming the user didn't change anything and clicked OK, $SRok=1 and $mixed=0xFF7F0000.

or:

```
$color:=0x80FF0000  // 50% transparent red
$mixed:=2
$SRok:=SR_ColorPicker($color;$mixed)  // return the mixed value
```

$SRok=1 and $mixed=0xFF7F0000.

This is used in SuperReport Pro 4D forms to display semi-transparent colors (4D does not support alpha channel).

SR_DetokenizeScript

(blob:O) → text:T

Parameter	Type	Description
→ blob	blob	Tokenized script in a blob.
← text	text	The detokenized script.

Detokenize a script that was previously tokenized with SR_TokenizeScript.

SR_ExecuteScript

(areaReportRef:L) → error:L

Parameter	Type	Description
→ areaReportRef	longint	Area / Report reference.

Set a flag to execute the object script at the next idle event.

Example

```
$error:=SR_ExecuteScript ($areaRepRef)
```

SR_RunScript

(text:T)

Parameter	Type	Description
➞ text	text	Script to execute.

Execute the script contained in **text**. The script can contain control structures. If you're familiar with the Footrunner plugin from Footprints, this command functions in very much the same way.

Example

Load a saved script from a record and execute it:

```
$script:=[Scripts]ScriptData
SR_RunScript ($script)
```

SR_RunTokenizedScript

(blob:O)

Parameter	Type	Description
➞ blob	blob	Tokenized script to execute.

Run a script that was tokenized with ***SR_TokenizeScript***.

SR_TokenizeScript

(text:T; blob:O) → length:L

Parameter	Type	Description
➞ text	text	Script to tokenize.
← blob	blob	The tokenized script.
← length	longint	Length of the blob.

Tokenize the script in **text** and save it into a **blob**.

If you need to create a long or complex script, you can create it in 4D, tokenize it using ***SR_TokenizeScript*** and store it into Resources: **Get 4D folder** (Current Resources folder).

You can then use **DOCUMENT TO BLOB** and run it with ***SR_RunTokenizedScript***.

Properties by Theme

In this section you'll find complete details about each property that can be used with the SuperReport Pro commands.

Property themes

They are organised into themes according to which Object Kinds they relate to:

- Common properties: Objects and Styles
  Object Common Properties
  Style Properties
- Plugin / Area / Event
  Plugin Properties
  Area Properties
  Event Properties
- Report
- Section / Guide
  Section General Properties
  Section Header / Footer Properties
  Section Break Properties
  Section Watermark Properties
  Guide Properties
- Group / Line / Oval / Rectangle
  Group Properties
  Line Properties
  Oval Properties
  Rectangle Properties
- Picture / Text
  Picture Properties
  Text Properties
- Variable / Field / Data source
  Variable Properties
  Field Properties

Property Table Columns

The following details are included for each property:

Constant: the name of the property that you type into the command.

Get: whether the constant can be used in Getter commands

Set: whether it can be used in Setter commands

Per: Persistent. If a property is persistent, it means that the property is saved with the area definition and will be applied when the area is displayed again.

Type: the type of the value:

Bool: boolean value (True=1 or False=0)

Int: a long integer

Real: a real number

Text: an alphanumeric

Color: the “Color” type will accept seven methods, whether as string values or longint values.See Working with colors.

Default: the default value that will be used for this property unless you specify otherwise

Min: the minimum acceptable value, where appropriate

Max: the maximum acceptable value, where appropriate

Comments: a description of the constant and, where appropriate, a list of allowable options

Note: property values and XML Names are listed in Appendix 4.

Common properties: Objects and Styles

These two groups include properties that may apply to various object kinds. In addition, the relevant properties for both are mentioned in the property list of each object kind to which they apply.

Object Common Properties

The following properties may or may not apply depending upon the objects (area, report, section, line, variable, table, etc.).

The characteristics below are common to all object types, except:

- The Object kind name.
- Some features that are different when applying to specific objects, indicated by a light purple background in the table below, see Comments for details.

SuperReport Pro Object Common Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Object_Kind	✔		✔	text				Depends upon the object kind See SRP Object Kinds constants
SRP_Object_XML	✔	✔	✔	text				XML of the object Report object: read only Use SR_LoadReport to set a new report Table header/footer objects: only when visible (not covered by another cell)
SRP_Object_ID	✔	✔	✔	text				Object ID used in XML/HTML export Style object: uses longint ID Guide object: not persistent, no XML name
SRP_Object_Name	✔	✔	✔	text				Object name used in XML/HTML export Style object: name used in styles popup Report object: name used in XML/HTML export/print job name
SRP_Object_Order	✔	✔	✔	int				Print order
SRP_Object_Rect	✔	✔		rect				Object rectangle as "left;top;right;bottom"- relative to the parent
SRP_Object_Visible	✔	✔	✔	bool	yes			Object is visible (is not hidden) Table header/footer objects: read only (not applicable to Table column) Guide object: not persistent, no XML name
SRP_Object_Locked	✔	✔	✔	int	0	0	2	0 = unlocked 1 = locked 2 = locked & not selectable Report object: not persistent, no XML name Guide object: not persistent, no XML name
SRP_Object_Selected	✔	✔	✔	bool	no			Object is selected in the report design editor Style object: not persistent, no XML name Report object: not persistent, no XML name Data Source object: not persistent, no XML name Guide object: not persistent, no XML name
SRP_Object_Export	✔	✔	✔	bool	yes			Should be exported? Note: line, oval and rectangle objects are never exported

SuperReport Pro Object Common Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Object_PosLeft	✔	✔	✔	real				Left coordinate of the object position - relative to the parent Table header / column / footer objects: read only
SRP_Object_PosTop	✔	✔	✔	real				Top coordinate of the object position - relative to the parent Table header / column / footer objects: read only
SRP_Object_PosRight	✔	✔		real				Right coordinate of the object position - relative to the parent
SRP_Object_PosBottom	✔	✔		real				Bottom coordinate of the object position - relative to the parent
SRP_Object_PosWidth	✔	✔	✔	real				Width of the object position - relative to the parent Table header / column / footer objects: not persistent, no XML name
SRP_Object_PosHeight	✔	✔	✔	real				Height of the object position - relative to the parent Table header / column / footer objects: not persistent, no XML name
SRP_Object_FixH	✔	✔	✔	bool	no			Fixed horizontal position
SRP_Object_FixV	✔	✔	✔	bool	no			Fixed vertical position
SRP_Object_VariableSizeH	✔	✔	✔	bool	no			Can grow / shrink horizontally
SRP_Object_VariableSizeV	✔	✔	✔	bool	no			Can grow / shrink vertically
SRP_Object_BindToParentH	✔	✔	✔	int	0	0	2	Bind to right side of parent: 0 = none SRP_Object_Bind_None 1 = move SRP_Object_Bind_Move 2 = grow SRP_Object_Bind_Grow
SRP_Object_BindToParentV	✔	✔	✔	int	0	0	2	Bind to bottom side of parent: 0 = none SRP_Object_Bind_None 1 = move SRP_Object_Bind_Move 2 = grow SRP_Object_Bind_Grow
SRP_Object_Align	✔	✔	✔	int	0	0	3	Position in parent 0 = current 1 = left 2 = center 3 = right Every relevant object in a section can be positioned on the left, in the middle or on the right This is usable when the report is set to physical paper (SRP_Report_PhysicalPaper) - it takes care of margins Otherwise (when the report is set to printable area) align right will be at the right margin
SRP_Object_Draw	✔	✔	✔	int	1	0	3	Draw object: 0 = no 1 = yes 2 = on overflow 3 = always Section object: uses two-state (boolean) values See Overflow
SRP_Object_RelPosLeft		✔		real				Move left side by X points Table header / column / footer objects: read only

SuperReport Pro Object Common Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Object_RelPosTop		✔		real				Move top side by X points Table header / column / footer objects: read only
SRP_Object_RelPosRight		✔		real				Move right side by X points
SRP_Object_RelPosBottom		✔		real				Move bottom side by X points
SRP_Object_RelMoveH		✔		real				Move horizontally by X points
SRP_Object_RelMoveV		✔		real				Move vertically by X points
SRP_Object_DashStyle	✔	✔	✔	int	0	0	4	Dash pattern for stroking lines 0 = solid —————— 1 = dash — — — — 2 = dot ..... 3 = dash dot —.—.—.— 4 = dash dot dot —..—..—.. See http://msdn.microsoft.com/en-us/library/windows/desktop/ms534104(v=vs.85).aspx Note: supported by Line, Oval, Rect, Picture, Text, Variable, Field
SRP_Object_Fill	✔	✔	✔	bool	no			Use the fill/background color
SRP_Object_Script	✔	✔	✔	text				Script to execute when the object is printed
SRP_Object_HTMLPrefix	✔	✔	✔	text				Used with HTML export
SRP_Object_HTMLSuffix	✔	✔	✔	text				Used with HTML export
SRP_Object_Frame	✔	✔	✔	int	0	0	1	The object has a visible frame Oval and Table objects: default is 1 Table object: maximum is 2 (single / double frame) Note: Rectangle objects use SRP_Rect_Flags
SRP_Object_FrameOffset	✔	✔	✔	real	2	0	32	Offset on inside of the frame in points For example, if a text object has its height set to 12, SRP_Object_Frame is on, and the SRP_Object_FrameOffset is 2, there will be only 8 points for the object text To use a frame offset with no frame, set SRP_Object_Frame to Yes (1) and SRP_Object_FrameThickness to 0
SRP_Object_FrameThickness	✔	✔	✔	real	1	0	10	Width of the frame in points

Style Properties

The following properties are used by text type objects.

Their purpose is to define styles that can later be used by relevant object kinds. They will also appear in the design editor under the **Styles** popup menu.

Examples

Here are examples of how to use styles:

Style Creation (using commands)

```
C_LONGINT($objNum)  // style object number
$objNum:=0  // will be modified by SR_NewObject below
$error:=SR_NewObject ($areaRepRef;$objNum;SRP_Style;0)  // create a new style, object number is $objNum
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_ID;123)  // set the style object ID to 123 (longint)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Object_Name;"My Style #123")  // style name
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Style_FontName;"Times")
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Size;12)
SR_SetTextProperty ($areaRepRef;$objNum;SRP_Style_TextColor;"red")
 // style "My Style #123" is now Times 12 Red
```

Style Creation (using XML)

```
C_LONGINT($objNum)  // style object number
$objNum:=0  // will be modified by SR_NewObjectFromXML below
$error:=SR_NewObjectFromXML ($areaRepRef;$objNum;"<Style name=\"My Style #123\" id=\"123\"\
   font=\"Times\" size=\"12\" textColor=\"red\" />")  // create and initialize a new style
```

Warning: Style ID Uniqueness

SuperReport Pro does not check for uniqueness in object IDs. Make sure that the style ID that you set does not already exist, otherwise the first created style under that ID will be used when applying the style.

This check can easily be performed with the ***SR_FindObjectByID*** command, which will return zero in the (style) object number and an error -5 = SRP_Err_InvalidObjectRef if this ID doesn't yet exist, meaning that the value can be added - it will be unique. Note that the ID is converted to text for the command:

```
C_LONGINT($objNum)  // style object number
$objNum:=0
$error:=SR_FindObjectByID ($areaRepRef;"123";$objNum)  // does any object exist with ID 123?
If ($objNum=0)  // $error = -5 = SRP_Err_InvalidObjectRef
 // The ID is unique, we can create the style
End if
```

Applying the Style to an Object

```
$styleID:=123  //we want to apply style #123 created above
SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID;$styleID)
 //object #$objNum is now Times 12 Red
```

See also Creating Reports Procedurally.

Additional Notes

- When the specified style is not available, the Default style (ID = 0) is used.
- In the design editor, when pasting a style which already exists, the old one is replaced.
- When you programmatically set a style ID to be duplicate, you can later change it with SRP_Object_ID

Common Object Properties used in Styles

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Style
SRP_Object_XML	
SRP_Object_ID	Type: int Style ID
SRP_Object_Name	
SRP_Object_Selected	Not persistent

Properties

SuperReport Pro Style Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Object_StyleID	✔	✔	✔	int	0	0		Style ID (set by SRP_Object_ID with SRP_Object_Kind = "Style" Base Style to use for the object Longint ID for Styles as opposed to text for all other object kinds Note: when set, all overriden (specific) properties are ignored until SRP_Style_Features is used to restore any of them See Note about SRP_Object_StyleID and Style Features
SRP_Style_Features	✔	✔	✔	int	1	1		Specifically defined (overriden) properties from the predefined style Useful to find out which features were modified since the style was set, through programming or with the design editor, thus differ from the style settings The value is a combination of feature bits - see Style Features constants
SRP_Style_FontName	✔	✔	✔	text	Arial on Windows Helvetica on MacOS			Style font name

SuperReport Pro Style Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Style_Size	✔	✔	✔	real	8 on Windows 9 on MacOS	4	128	Style font size Type is real because you can use decimal values such as 10.5
SRP_Style_Full	✔	✔		int	0	0	15	Style font style, combining the values of the following boolean properties (each assigned to a bit): SRP_Style_Bold (bit 0) SRP_Style_Italic (bit 1) SRP_Style_Underline (bit 2) SRP_Style_StrikeThrough (bit 3)
SRP_Style_Bold	✔	✔		bool	no			Style font style - bold
SRP_Style_Italic	✔	✔		bool	no			Style font style - italic
SRP_Style_Underline	✔	✔		bool	no			Style font style - underline
SRP_Style_StrikeThrough	✔	✔		bool	no			Style font style - strike-through
SRP_Style_Wrap	✔	✔	✔	bool	no			Style wrap long lines
SRP_Style_HorAlign	✔	✔	✔	int	0	0	5	Style horizontal alignment: 0 = default 1 = left 2 = center 3 = right 4 = justify 5 = full justify
SRP_Style_VertAlign	✔	✔	✔	int	0	0	3	Style: vertical alignment: 0 = default 1 = top 2 = center 3 = bottom
SRP_Style_TextColor	✔	✔	✔	color	#FF000000			Style: font color Default is Black
SRP_Style_BackColor	✔	✔	✔	color	#00FFFFFF			Style: background color Default is Transparent (no color)
SRP_Style_FrameColor	✔	✔	✔	color	#FF000000			Style: frame color Default is Black
SRP_Style_Rotation	✔	✔	✔	real	0	-360	360	Style rotation of text
SRP_Style_BaseLineShift	✔	✔	✔	real	0	-100	256	Style baseline shift
SRP_Style_HorizontalScale	✔	✔	✔	real	1	0.1	100	Style horizontal scale
SRP_Style_LineSpacing	✔	✔	✔	real	1	0.5	10	Style line spacing Default value can be changed using ***SR_SetRealProperty*** (0;0;SRP_Style_LineSpacing;$value)
SRP_Style_FirstLineIndent	✔	✔	✔	real	0	-256	256	Negative is “hanging indent” Positive is “first line indent” New “paragraph” is determined by new line (carriage return)

Note: changing the value to any of the five Style font style properties (full, bold, italic, underline, strike-through) will modify only one SRP_Style_Features bit: SRP_Style_HasFontStyle (bit 3).

Note about SRP_Object_StyleID and SRP_Style_Features

When you apply a base style to a text object with SRP_Object_StyleID the “overridden” or “specific” properties a.k.a. “features” (specifically set to the object through any of the other Style properties) are no longer applied to the object.

The SRP_Style_Features property is set to 1 (SRP_Style_HasBaseStyle). When you access any style property thereafter, the base style is used (until any property is overriden).

You can set any bit of SRP_Style_Features explicitly: the appropriate last known feature (“specific” value) will be used.

However, this feature restore ability is only valid during the current editing session. Features that are no longer used (were reset to base style values) are not saved with the report. In other words, only visible features (effectively in use) are saved with the report.

Example:

start with a virgin object using default style (styleID = 0)

features = 1 (SRP_Style_HasBaseStyle)

fontName = (e.g.) "Helvetica"

textColor = "#FF000000"

set fontName to "Arial"

features = 3 (SRP_Style_HasBaseStyle | SRP_Style_HasFontName)

fontName = "Arial"

textColor = "#FF000000"

set textColor to "red"

features = 19 (SRP_Style_HasBaseStyle | SRP_Style_HasFontName | SRP_Style_HasTextColor)

fontName = "Arial"

textColor = "#FFFF0000"

set styleID to 0

features = 1 (SRP_Style_HasBaseStyle)

fontName = (e.g.) "Helvetica"

textColor = "#FF000000"

set feature to 17

features = 17 (SRP_Style_HasBaseStyle | SRP_Style_HasTextColor)

fontName = (e.g.) "Helvetica"

textColor = "#FFFF0000"

When saved, the report XML will only include textColor = "#FFFF0000" as a specific feature. The fontName = "Arial" feature that was reset will be forgotten.

Plugin/Area/Event

Plugin Properties

Note: these properties expect the value zero as the first two parameters to the getter/setter command.

Style Property used

- SRP_Style_LineSpacing

Properties

SuperReport Pro Plugin Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_Version	✔			text				Version of the SuperReport Pro plugin
SRP_Area_Path	✔			text				Path to the SuperReport Pro plugin
SRP_Area_TraceOnError	✔	✔		int	1	0	3	Invoke the 4D debugger in interpreted and/or an alert in compiled if a command causes an error, and it is a command that does not return an error code bit 0: trace in interpreted (values 1 & 3) bit 1: alert in compiled (values 2 & 3)
SRP_Area_LastError	✔	✔		int				Last error in **any** SuperReport Pro report or area
SRP_Area_Copyright	✔			text				Copyright of the SuperReport Pro plugin
SRP_Area_Rounding	✔	✔		real	1	0.1	10000	If set to N, all position values are rounded to 1/N of point For example, setting it to 100 will cause all positions be rounded to 1/100 (0.01) point Used in the SuperReport Pro design editor
SRP_Area_Proximity	✔	✔		real	2.5	0.05	10	Distance from which will be dragged points snapped to guides or grid points Used in the SuperReport Pro design editor
SRP_Area_ScrollWheel	✔	✔		real	10	1	1000	Multiplier on mouse wheel move If you use the mouse wheel: - the option key will make the scroll 10 times faster - the control/command key will make the scroll SRP_Area_ScrollWheel times faster Used in the SuperReport Pro design editor
SRP_Area_NewReport	✔			text				Default empty report Stored in Resources/EmptyReport.XML
SRP_Area_VarDate	✔	✔		text	SRDate			Standard variable name
SRP_Area_VarTime	✔	✔		text	SRTime			Standard variable name
SRP_Area_VarPage	✔	✔		text	SRPage			Standard variable name
SRP_Area_VarRecord	✔	✔		text	SRRecord			Standard variable name
SRP_Area_VarArea	✔	✔		text	SRArea			Standard variable name

SuperReport Pro Plugin Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_VarObject	✔	✔		text	SRObjectPrintRef			Standard variable name SRObjectID is maintained for compatibility, but only the name SRObjectPrintRef can be modified with this property
SRP_Area_VarBegHTML	✔	✔		text	SRBegHTML			Standard variable name
SRP_Area_VarEndHTML	✔	✔		text	SREndHTML			Standard variable name
SRP_Area_VarPages	✔	✔		text	SRPages			Standard variable name
SRP_Area_VarName	✔	✔		text	SRName			Standard variable name Report only variable (not a 4D variable)
SRP_Area_VarDateTime	✔	✔		text	SRDateTime			Standard variable name Report only variable (not a 4D variable)
SRP_Area_VarPrintSection	✔	✔		text	SRPrintSection			Standard variable name
SRP_Area_VarCurrentRun	✔	✔		text	SRCurrentRun			Standard variable name
SRP_Area_VarRepeat	✔	✔		text	SRRepeatNum			Standard variable name Only valid in scripts of repeating objects
SRP_Area_ToolTips	✔	✔		int		0	2	Enable tool tips 0 = no 1 = yes 2 = toggle
SRP_Area_ExtensionSRP	✔	✔		text	XML			Standard file extension for report Old ".srp" is always accepted for compatibility in addition to this setting Setting an empty string resets the extension to the default
SRP_Area_ExtensionTXT	✔	✔		text	txt			Standard file extension for text export Setting an empty string resets the extension to the default
SRP_Area_ExtensionHTML	✔	✔		text	html			Standard file extension for HTML export Setting an empty string resets the extension to the default
SRP_Area_ExtensionXML	✔	✔		text	XML			Standard file extension for XML export Setting an empty string resets the extension to the default
SRP_Area_ NotificationCallback	✔	✔		text				Notification callback (e.g. for properties palette implementation) Parameters: (AreaReportRef:L; CallbackType:L; Rebuild:B) - CallbackType is the same as in event or editor callbacks, see SRP_Area_CallbackMethEdit - Rebuild is True if the report was modified since the last call
SRP_Area_PreviewFlags	✔	✔		int	0			Additional flags (bits) to use when preview in the editor is used or old v2 ***SR Preview*** is called See Preview flags

SuperReport Pro Plugin Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_ WinPreviewName	✔	✔		text	""			Name of a file to use when Preview is used on Windows Preview means SRP_Print_DestinationPreview is specified (preview in editor, ***SR Preview***, SR_Print) See also Print Flags
SRP_Area_Moving	✔	✔		real	1	0.1	5	Step for moving / resizing objects with arrow keys Values are in points (decimals allowed) Also settable in Report Properties dialog

Area Properties

Common Object Property used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Area

Properties

SuperReport Pro Area Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_Name	✔			text				Name of the area object (variable name on 4D form)
SRP_Area_IsArea	✔			bool				Is this a SuperReport Pro area?
SRP_Area_Visible	✔	✔		bool				Area is visible Set to false before showing another dialog over the SuperReport Pro area to hide scrollbars
SRP_Area_Selected	✔			bool				Is selected (has focus in 4D)
SRP_Area_ScrollLeft	✔	✔		real				Horizontal scroll position in points (decimals allowed)
SRP_Area_ScrollTop	✔	✔		real				Vertical scroll position in points (decimals allowed)
SRP_Area_Self	✔			pointer				Pointer to the area variable **C_POINTER**($ptr) $error:=***SR_GetPtrProperty*** ($areaRepRef;1;SRP_Area_Self;->$ptr)

SuperReport Pro Area Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_Tool	✔	✔		int	0	0	8	Current tool: 0 = selection / SRP_Tool_Select 1 = text / SRP_Tool_Text 2 = field / SRP_Tool_Field 3 = variable / SRP_Tool_Variable 4 = line / SRP_Tool_Line 5 = rectangle / SRP_Tool_Rectangle 6 = oval / SRP_Tool_Oval 7 = picture / SRP_Tool_Picture 8 = table / SRP_Tool_Table See SRP Area Tools constants
SRP_Area_DrawingPage	✔	✔		int	0	0	3	Draw sections as of page X 0 = all 1 = first page 2 = second page 3 = last page
SRP_Area_DrawingMode	✔	✔		int	0	0	7	Drawing mode of the objects: 0 = normal 1 = alias 2 = format 3 = name 4 = ID 5 = order 6 = size 7 = value
SRP_Area_Report	✔			int				Report reference to the associated report
SRP_Area_Redraw		✔		n/a				Force the area redraw

Event Properties

SuperReport Pro Event Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_Event	✔			int				Kind of event: 1 = mouse down 3 = key down 5 = auto key 18 = mouse moved 21 = select 22 = deselect 25 = scroll 30 = Undo 31 = Cut 32 = Copy 33 = Paste 34 = Clear 35 = Select all 36 = Redo 39 = mouse wheel
SRP_Area_EventPosH	✔			int				Horizontal mouse position in points
SRP_Area_EventPosV	✔			int				Vertical mouse position in points
SRP_Area_EventModifiers	✔			int				Event modifiers: 256 = command 512 = shift 1024 = caps lock 2048 = option 4096 = control
SRP_Area_EventKey	✔			text				Key code
SRP_Area_EventChar	✔			text				Char (string) from keyboard
SRP_Area_DoubleClick	✔			bool				Last click is double click

SuperReport Pro Event Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_ CallbackMethEdit	✔	✔	✔	text				Editor event callback function Parameters: (AreaReportRef:L; CallbackType:L; ObjectNum:L; ObjectType:L) -> Result:L CallbackType values: 1 = object created 2 = object modified 3 = section modified 4 = object script modified 5 = report script modified 6 = control-click 7 = click 8 = selection changed 9 = object deleted ObjectType values: 0 = group 1 = line 2 = rectangle 3 = oval 4 = picture 5 = text 6 = variable 7 = field 8 = table 9 = table header 10 = table column 11 = table footer 12 = report document 13 = report area 14 = data source 15 = style 16 = section 17 = guide Note that the values above are not compatible with pre-v3 versions Result values: 0 = handled 1 = not handled, SuperReport Pro should proceed with handling

SuperReport Pro Event Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Area_HitPart	✔			int				Part of object under mouse: 0 = none 1 = object 2 = top / left 3 = top / center 4 = top / right 5 = right / center 6 = bottom / right 7 = bottom / center 8 = bottom / left 9 = left / center 10 = horizontal resize 11 = vertical resize
SRP_Area_HitObject	✔			int				Object under mouse
SRP_Area_CreatedObject	✔	✔		int				Last object created by user Setter clears this property
SRP_Area_DraggedObject	✔			int				Object under the mouse during drag
SRP_Area_DraggedPosH	✔			int				Horizontal position of the mouse pointer during drag
SRP_Area_DraggedPosV	✔			int				Vertical position of the mouse pointer during drag

Report

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Report
SRP_Object_XML	Read only: use SR_LoadReport to set a new report
SRP_Object_ID	
SRP_Object_Name	Name used in XML / HTML export / print job name
SRP_Object_Locked	Not persistent, no XML name Make it locked, hide toolbar & menubar to get read-only view
SRP_Object_Selected	Not persistent, no XML name

Report Properties

SuperReport Pro Report Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Report_Version	✔		✔	text				Version of the report format
SRP_Report_PageWidth	✔	✔	✔	real		100	32767	The page width in points
SRP_Report_PageHeight	✔	✔	✔	real		100	32767	The page height in points
SRP_Report_Scale	✔	✔	✔	real	1	0.1	10	Printing scale The scaling is provided by SuperReport Pro, not by the printer driver
SRP_Report_PhysicalPaper	✔	✔	✔	bool	no			Use physical paper Printable area otherwise (default)
SRP_Report_Margins	✔	✔		text				Page margins: left, top, right, bottom as a string e.g.: "1.5;1.5;1.5;1.5"
SRP_Report_MarginLeft	✔	✔	✔	real		0	256	Left-hand margin in points
SRP_Report_MarginTop	✔	✔	✔	real		0	256	Top margin in points
SRP_Report_MarginRight	✔	✔	✔	real		0	256	Right-hand margin in points
SRP_Report_MarginBottom	✔	✔	✔	real		0	256	Bottom margin in points
SRP_Report_CountPages	✔	✔	✔	bool	no			Calculate number of pages before printing If not set, SRPages will contain -1
SRP_Report_ColumnCount	✔	✔	✔	int	1	1	32	Number of columns in the Body section
SRP_Report_ ColumnSpacing	✔	✔	✔	real	5	0	256	Space between the columns in points
SRP_Report_ColumnOrder	✔	✔	✔	bool	no			Column print order is horizontal
SRP_Report_PageFormat	✔	✔	✔	BLOB				MacOS: flattened page format (XML)
SRP_Report_PrintSettings	✔	✔	✔	BLOB				MacOS: flattened print settings (XML)
SRP_Report_DevMode	✔	✔	✔	BLOB				Windows: device mode (binary)
SRP_Report_DeviceNames	✔	✔	✔	BLOB				Windows: device names (binary)

SuperReport Pro Report Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Report_ ObjectHierarchy	✔			BLOB				All sections with all objects in the report as List in blob Use **BLOB TO LIST** to get the list object
SRP_Report_UserBLOB	✔	✔		BLOB				BLOB for free use by developer
SRP_Report_Document	✔	✔		text				Full path to the document
SRP_Report_DataSource	✔			int				Object number of the data source
SRP_Report_MacPrinter	✔		✔	text				MacOS: device name (from the page setup XML stored in the SRP_Report_PageFormat property)
SRP_Report_WinPrinter	✔		✔	text				Windows: device name (from the binary DEVNAMES structure stored in the SRP_Report_DeviceNames property)
SRP_Report_PrintEmptyText	✔	✔	✔	bool	no			Set to 1 (True) to override old v2.x behavior and print empty text Note: when this property is set to 1 (True), SRP_ Text_PrintEmptyText, SRP_Variable_PrintEmptyText and SRP_Field_PrintEmptyText are ignored (handled as if set to True)

Report Editor Properties

SuperReport Pro Report Editor Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Report_ CallbackMethEvent	✔	✔	✔	text				Event callback method Parameters: (AreaReportRef:L; Event:L; Info:L) -> Result:L Event values: 11 = area zoomed (parameter 3 contains the area reference in the external window) 12 = zoomed window collapsed (un-zoomed) 13 = zoomed window brought to front 14 = original (4D form) window brought to front 15 = editor window has been closed (either report area on form or external window) 16 = area has lost focus 17 = area is hidden (changing form page) 40 = menu item selected before (parameter 3 contains the menu item ID) 41 = menu item selected after (parameter 3 contains the menu item ID) If SRP_Report_CallbackVersion is set to 1, a return value is expected for “before menu” event (40): return 1 to execute menu action

SuperReport Pro Report Editor Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**

SRP_Report_ CallbackVersion	✔	✔		int	0	0	1	Callback version of the Menu handler to use
								Different parameters are used when calling the method to be executed for a menu item
								0 = old behavior: called without parameters
								1 = new behavior: called with parameters (area; menuID)
								If set to 1, modify behavior of menu event callback and all custom methods associated with menu items
								If set to 1, menu event callback (set with SRP_Report_CallbackMethEvent or old 'SR On Event') has to add:
								C_LONGINT($0)
								$0:=1 // set to 0 to not execute the menu item method / standard action
								And any method called as menu item override (set with ***SR Menu Item*** ($areaRepRef; SR MenuItem Set 4D Method; $menuID; "";0;0; "myMenuItemCallback")) has to add:
								C_LONGINT($1) // the SRP area
								C_LONGINT($2) // menu ID
								(required to function in compiled mode)
								This is how you can have one method for all menu item action overrides

SRP_Report_ShowMenubar	✔	✔		bool	yes			Display the SuperReport Pro menu bar
SRP_Report_ShowToolbar	✔	✔		bool	yes			Display the SuperReport Pro toolbar
SRP_Report_ShowMargins	✔	✔	✔	bool	yes			Display the margins
SRP_Report_ShowRuler	✔	✔	✔	bool	no			Display the SuperReport Pro ruler
SRP_Report_RulerUnits	✔	✔	✔	int	1	1	3	1 = point 2 = mm 3 = inch
SRP_Report_GridSize	✔	✔	✔	int	12	4	256	Grid size, in SRP_Report_RulerUnits
SRP_Report_ShowGrid	✔	✔	✔	bool	no			Display the grid
SRP_Report_SnapToGrid	✔	✔	✔	bool	no			Snap objects to the grid
SRP_Report_ShowGuides	✔	✔	✔	bool	yes			Snap objects to the guides
SRP_Report_LockGuides	✔	✔	✔	bool	no			Lock the guides
SRP_Report_SnapToGuides	✔	✔	✔	bool	no			Snap objects to the guides
SRP_Report_ShowSections	✔	✔	✔	bool	yes			Show section labels
SRP_Report_LockSections	✔	✔	✔	bool	no			Lock the sections of the report
SRP_Report_ ShowObjBorders	✔	✔	✔	bool	no			Show objects' borders
SRP_Report_Zoom	✔	✔	✔	real	1	0.1	10	View scale
SRP_Report_ShowZoom	✔	✔		bool	yes			Enable Zoom (open report in external window)
SRP_Report_ShowSRMenu	✔	✔		bool	yes			Show the Database menu
SRP_Report_BasicSRMenu	✔	✔		bool	no			Show the basic dialog on **Database** menu > **Main Table**
SRP_Report_EnableScripts	✔	✔		bool	yes			Enable editing of scripts

SuperReport Pro Report Editor Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Report_ExportFlags	✔	✔		int	0			Flags used with **File** > **Export** (last set by user): 256 = SRP_Export_Text 16 = SRP_Export_CSVFormat 512 = SRP_Export_XML 1024 = SRP_Export_HTML 4096 = SRP_Export_Body 8192 = SRP_Export_Breaks 16384 = SRP_Export_Total 32768 = SRP_Export_Headers 65536 = SRP_Export_Watermark 32 = SRP_Export_StaticText 64 = SRP_Export_Sorted 128 = SRP_Export_PlainText
SRP_Report_ ExportDelimiter	✔	✔		int	TAB (9)			The field delimiter for text export used with **File** > **Export**
SRP_Report_ ExportRecDelimiter	✔	✔		int	CR+LF (655373)			Record delimiter to be used for text export This value is a long integer which can contain 2 characters where the first character is put in the low-order word The default is "\r\n" which is equivalent to Carriage Return | (Line Feed << 16) The record delimiter is always CR+LF in XML and HTML mode
SRP_Report_DoMenu		✔		int				Execute specified menu item (or associated method if specified for that menu item)
SRP_Report_ DoMenuNoProc		✔		int				Execute specified menu item (ignoring associated method if specified for that menu item)
SRP_Report_HideHTML	✔	✔		bool				Hide HTML in User Interface
SRP_Report_Modified	✔	✔		bool				True (1) if modified
SRP_Report_ZoomTitle	✔	✔		text	SuperReport Pro			Name to use when “zooming” the area to an external window
SRP_Report_ EditNumberPages	✔	✔	✔	int	1	1	16	Number of pages to draw in the editor Usable for designing multipage reports

Section / Guide

Section Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Section
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_RelPosBottom	
SRP_Object_Draw	Uses two-state (boolean) values
SRP_Object_Script	
SRP_Object_HTMLPrefix	
SRP_Object_HTMLSuffix	

Section General Properties

SuperReport Pro Section General Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_Type	✔		✔	text				SRP_SectionType_Header SRP_SectionType_BreakHeader SRP_SectionType_Body SRP_SectionType_BreakFooter SRP_SectionType_Footer SRP_SectionType_Watermark See SRP Section Types “Total” is SRP_SectionType_BreakFooter with break level = 0
SRP_Section_Height	✔	✔	✔	real	0	0	4096	Height of the section in points
SRP_Section_FixedHeight	✔	✔	✔	bool	no			The section will not grow / shrink if set to true; Footer always has fixed height
SRP_Section_KeepTogether	✔	✔	✔	bool	no			Start a new page if there is not enough space for the entire section
SRP_Section_MinSpace	✔	✔	✔	real	0	0	512	Minimum available space on page; if less is available, start a new page

SuperReport Pro Section General Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_PageThrow	✔	✔	✔	int	0	0	2	Throw a new page: 0 = none 1 = before the section is printed 2 = after the section is printed

Section Header/Footer Properties

You can have 3 separate headers/footers, each specifying only one of the three options - then you have separate first page header, last page header, header printed on other pages.

SuperReport Pro Section Header/Footer Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_FirstPage	✔	✔	✔	bool	yes			Print on first page
SRP_Section_SecondPage	✔	✔	✔	bool	yes			Print on subsequent pages except the last page
SRP_Section_LastPage	✔	✔	✔	bool	yes			Print on last page

Section Break Properties

SuperReport Pro Section Break Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_ FooterFromBottom	✔	✔	✔	bool	no			Print the section from bottom of page, not below the body This is a property of break footer and footer, where it defaults to true for footer (footer at the bottom of page) and false for break footer (break after the body/higher-level breaks, not above the footer)
SRP_Section_Break_ OnField	✔	✔	✔	text				These are “2 in 1” - when set, SRP_Section_Break_Type is set and SRP_Section_Break_OnXXX is set to the provided source (variable name or [Table]Field) When a getter is used, only the “correct” one (according to SRP_Section_Break_Type) returns SRP_Section_Break_OnXXX (the variable/field to check for value change)
SRP_Section_Break_ OnVariable	✔	✔	✔	text				
SRP_Section_Break_ OnArray	✔	✔	✔	text				
SRP_Section_Break_Level	✔	✔	✔	int	0	0		Section break level 0 = total
SRP_Section_Break_Always	✔	✔	✔	bool	no			Always draw on new page Break header only
SRP_Section_Break_On	✔	✔		text				The variable/field to check for value change

SuperReport Pro Section Break Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_Break_Type	✔	✔		int	0	1	3	The type of break: 0 = none 1 = field 2 = variable 3 = array {currentIteration}

Section Watermark Properties

SuperReport Pro Section Watermark Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Section_FirstPage	✔	✔	✔	bool	yes			Print the watermark on the first page
SRP_Section_SecondPage	✔	✔	✔	bool	yes			Print the watermark on subsequent pages except the last page
SRP_Section_LastPage	✔	✔	✔	bool	yes			Print the watermark on the last page
SRP_Section_Watermark_OnTop	✔	✔	✔	bool	no			Print on top of all other drawings Note: when this property is 0, the objects in the watermark section are not selectable with the mouse

Guide Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Guide
SRP_Object_XML	
SRP_Object_ID	Not persistent, no XML name
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	Not persistent, no XML name
SRP_Object_Locked	Not persistent, no XML name
SRP_Object_Selected	Not persistent, no XML name
SRP_Object_RelPosLeft	Only vertical guide
SRP_Object_RelPosTop	Only horizontal guide
SRP_Object_RelMoveH	Only vertical guide
SRP_Object_RelMoveV	Only horizontal guide

Properties

SuperReport Pro Guide Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Guide_Type	✔		✔	bool				0 = horizontal 1 = vertical
SRP_Guide_Position	✔	✔	✔	real	0	-4096	4096	Location of the guide

Group / Line / Oval / Rectangle

Group Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Group
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	

Constant	Specifics
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	

Properties

SuperReport Pro Group Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Group_Objects	✔			int				Number of objects owned by this group
SRP_Group_KeepTogether	✔	✔	✔	bool	no			“Keep” together a group of objects

Line Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Line
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	

Constant	Specifics
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	

Properties

SuperReport Pro Line Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Line_Thickness	✔	✔	✔	real	1	0	10	Line thickness in points
SRP_Line_Color	✔	✔	✔	color	black			Line color See Working with colors
SRP_Line_Flags	✔	✔	✔	int	undefined	0	4	Line kind: 0 = horizontal 1 = vertical 2 = top/left to bottom/right 3 = bottom/left to top/right 4 = full X

Oval Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Oval
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	

Constant	Specifics
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	
SRP_Object_Frame	Default is 1

Properties

SuperReport Pro Oval Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Oval_Thickness	✔	✔	✔	real	1	0	10	Thickness of the oval's frame, in points
SRP_Oval_Color	✔	✔	✔	color	black			Color of the frame See Working with colors
SRP_Oval_FillColor	✔	✔	✔	color	black			Color to fill the oval with See Working with colors

Rectangle Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Rectangle
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	

Properties

SuperReport Pro Rectangle Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Rect_Thickness	✔	✔	✔	real	1	0	10	Width of the border in points
SRP_Rect_Color	✔	✔	✔	color	black			Color of the border See Working with colors
SRP_Rect_FillColor	✔	✔	✔	color	black			Color to fill the rectangle with See Working with colors
SRP_Rect_Rows	✔	✔	✔	int	1	1	512	Number of rows in the grid
SRP_Rect_Columns	✔	✔	✔	int	1	1	512	Number of columns in the grid
SRP_Rect_Flags	✔	✔	✔	int	15	0	31	Which border lines to draw: bit 0: left bit 1: top bit 2: right bit 3: bottom bit 4: rounded rect When a rounded rectangle is used, all four lines are drawn

Picture / Text

Picture Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Picture
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	
SRP_Object_Frame	
SRP_Object_FrameOffset	
SRP_Object_FrameThickness	

Properties

SuperReport Pro Picture Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Picture_FillColor	✔	✔	✔	color	black			Color with which to fill in the white spaces of the picture See Working with colors
SRP_Picture_FrameColor	✔	✔	✔	color	black			Color of the frame See Working with colors
SRP_Picture_HTMLReplace	✔	✔	✔	text				When exporting as HTML, put this text into the generated HTML in place of the actual picture Pictures can't be embedded into HTML; a link is needed
SRP_Picture_Data	✔	✔	✔	BLOB / PICT				The picture
SRP_Picture_Format	✔	✔	✔	int	0	0	4	0 = truncated 1 = centered 2 = scaled to fit 3 = scaled proportionally 4 = scaled proportionally centered
SRP_Picture_Size	✔			int				Image size
SRP_Picture_Width	✔			real				Width of the picture in points The picture must be displayed on screen, otherwise it will be 0
SRP_Picture_Height	✔			real				Height of the picture in points The picture must be displayed on screen, otherwise it will be 0

Text Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Text
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	

Constant	Specifics
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	
SRP_Object_HTMLPrefix	
SRP_Object_HTMLSuffix	
SRP_Object_Frame	
SRP_Object_FrameOffset	
SRP_Object_FrameThickness	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation
- SRP_Style_BaseLineShift
- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Text Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Text_Dynamic	✔	✔	✔	bool	no			Parse for variable substitution <%variable%>
SRP_Text_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Text_KeepTogether	✔	✔	✔	bool	no			If possible, keep all the text in the box on the same page If it does not fit into the available space, a new page is requested, then it is printed even if it does not fit on a page
SRP_Text_DrawEmpty	✔	✔	✔	int	0	0	2	0 = draw 1 = remove 2 = remove enclosing group
SRP_Text_Data	✔	✔	✔	text				The text
SRP_Text_ParsedData	✔			text				Parsed text with substituted variables When a View Values was used, the text/variable/ field is processed as at the time of printing - this is the result of processing the SRP_Text_Data (identical if SRP_Text_Dynamic is false)
SRP_Text_PrintEmptyText	✔	✔	✔	bool	no			Set to 1 (True) to override old v2.x behavior and print empty text

Variable / Field / Data source

Variable Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Variable
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	
SRP_Object_Script	
SRP_Object_HTMLPrefix	
SRP_Object_HTMLSuffix	
SRP_Object_Frame	
SRP_Object_FrameOffset	
SRP_Object_FrameThickness	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation
- SRP_Style_BaseLineShift
- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Variable Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Variable_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Variable_KeepTogether	✔	✔	✔	bool	no			Keep the entire contents of the frame on the same page
SRP_Variable_DrawEmpty	✔	✔	✔	int	0	0	2	0 = draw 1 = remove 2 = remove enclosing group
SRP_Variable_Source	✔	✔	✔	text				Variable to display
SRP_Variable_Index	✔	✔	✔	int	-2	-2		-2 = variable -1 = array {currentIteration} >=0 is array {index}
SRP_Variable_Alias	✔	✔	✔	text				Alias of the variable, as defined by the developer using the ***SR Variables*** command in the old SuperReport Pro API Only used in the editor
SRP_Variable_Format	✔	✔	✔	text				Formats as defined in 4D Pictures use formats as defined for picture objects, e.g. "3" for scaled proportionally For date/time variables, use 4D specific formats as text, e.g. "107" - can use **String** (Blank if null date + Internal date short)
SRP_Variable_ParsedData	✔			text				See SRP_Text_ParsedData, here it is the variable value (no <%variable%> substitution)
SRP_Variable_Calculate	✔	✔	✔	int	0	0	7	What to calculate: 0 = none 1 = total 2 = min 3 = avg 4 = max 5 = count 6 = stdvar 7 = stddev
SRP_Variable_Record	✔	✔	✔	text				Name of a variable to store the calculated value
SRP_Variable_Repeat	✔	✔	✔	int	0	0	2	Repeat this object: 0 = none 1 = horizontal 2 = vertical Note: consider using a Table object
SRP_Variable_RepeatOffset	✔	✔	✔	real	0	0	512	Offset (distance between) repeating objects in points
SRP_Variable_UseOld	✔	✔	✔	bool	yes			In break footer/total, use the old value (before the break)
SRP_Variable_PrintEmptyText	✔	✔	✔	bool	no			Set to 1 (True) to override old v2.x behavior and print empty text

Field Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Field
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_VariableSizeH	
SRP_Object_VariableSizeV	
SRP_Object_BindToParentH	
SRP_Object_BindToParentV	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_DashStyle	
SRP_Object_Fill	
SRP_Object_Script	
SRP_Object_HTMLPrefix	
SRP_Object_HTMLSuffix	
SRP_Object_Frame	
SRP_Object_FrameOffset	
SRP_Object_FrameThickness	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation
- SRP_Style_BaseLineShift
- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Field Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Field_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Field_KeepTogether	✔	✔	✔	bool	no			Keep the entire contents of the frame on the same page
SRP_Field_DrawEmpty	✔	✔	✔	int	0	0	2	0 = draw 1 = remove 2 = remove enclosing group
SRP_Field_Source	✔	✔	✔	text				The table and field to display
SRP_Field_Alias	✔	✔	✔	text				-2 = variable -1 = array {currentIteration} >=0 is array {index}
SRP_Field_Format	✔	✔	✔	text				Formats as defined in 4D Pictures use formats as defined for picture objects, e.g. "3" for scaled proportionally For date/time variables, use 4D specific formats as text, e.g. "107" - can use **String** (Blank if null date + Internal date short)
SRP_Field_ParsedData	✔			text				See SRP_Text_ParsedData, here it is the field value

SuperReport Pro Field Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Field_Calculate	✔	✔	✔	int	0	0	7	What to calculate: 0 = none 1 = total 2 = min 3 = avg 4 = max 5 = count 6 = stdvar 7 = stddev
SRP_Field_Record	✔	✔	✔	text				Name of a variable to store the calculated value
SRP_Field_Repeat	✔	✔	✔	int	0	0	2	Repeat this object: 0 = none 1 = horizontal 2 = vertical
SRP_Field_RepeatOffset	✔	✔	✔	real	0	0	512	Offset (distance between) repeating objects in points
SRP_Field_UseOld	✔	✔	✔	bool	yes			In break footer / total, use the old value (before the break)
SRP_Field_PrintEmptyText	✔	✔	✔	bool	no			Set to 1 (True) to override old v2.x behavior and print empty text

Data source Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_DataSource
SRP_Object_XML	
SRP_Object_Selected	Not persistent, no XML name

Properties

SuperReport Pro Data source Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_DataSource_Type	✔		✔	text	4D			For future enhancements, currently only "4D"
SRP_DataSource_Source	✔	✔	✔	text	0	1	4	What defines the number of iterations: 1 = records in selection of table (table number is in SRP_DataSource_TableID) 2 = fixed number specified by SRP_DataSource_Iterations 3 = value of variable (variable name is in SRP_DataSource_VarName) 4 = size of array (array name is in SRP_DataSource_VarName)
SRP_DataSource_VarName	✔	✔	✔	text				Variable name
SRP_DataSource_Alias	✔	✔	✔	text				Alias of the data source, as defined by the developer using the ***SR Variables*** command in the old SuperReport Pro API Only used in the editor
SRP_DataSource_Iterations	✔	✔	✔	int	1	1		Number of iterations
SRP_DataSource_TableID	✔	✔	✔	int	0			4D table number
SRP_DataSource_RelateOne	✔	✔	✔	int	2	0	2	What to do when SRP_DataSource_Source is 1 and fetching a record: 0 = relations are ignored 1 = use automatic relations 2 = use automatic relations & do a manual **RELATE ONE**/**RELATE MANY** on the main table (SRP_DataSource_TableID)
SRP_DataSource_RelateMany	✔	✔	✔	int	2	0	2	
SRP_DataSource_Callback	✔	✔	✔	text				Callback for scripts execution Parameters: (AreaReportRef:L; Script:T)
SRP_DataSource_StartScript	✔	✔	✔	text				Script to be executed before the report is processed Even before the number of iterations is checked (selection/array size/variable) - so you can create a selection for the report here
SRP_DataSource_BodyScript	✔	✔	✔	text				Script to be executed when a new row is fetched (going to the next iteration) This differs from the Body's script, which is executed just before processing the section (as all sections are handled the same) “Processing”, because it does not mean this section will be printed now - e.g. if there is not enough space: footer will be processed, then header, break headers…
SRP_DataSource_EndScript	✔	✔	✔	text				Script to be executed after the report is finished - to do some cleanup (e.g. clearing arrays created in the Start script for the report)
SRP_DataSource_StartScriptToks	✔	✔	✔	BLOB				Tokenized version of the script (SRP_DataSource_StartScript)
SRP_DataSource_BodyScriptToks	✔	✔	✔	BLOB				Tokenized version of the script (SRP_DataSource_BodyScript)
SRP_DataSource_EndScriptToks	✔	✔	✔	BLOB				Tokenized version of the script (SRP_DataSource_EndScript)

Table / Header / Column / Footer

Use these properties with Table objects.

Table Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Table
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	Read only
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_Export	
SRP_Object_PosLeft	
SRP_Object_PosTop	
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	
SRP_Object_PosHeight	
SRP_Object_FixH	
SRP_Object_FixV	
SRP_Object_BindToParentH	
SRP_Object_Align	
SRP_Object_Draw	
SRP_Object_RelPosLeft	
SRP_Object_RelPosTop	
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_RelMoveH	
SRP_Object_RelMoveV	
SRP_Object_StyleID	
SRP_Object_Script	
SRP_Object_HTMLPrefix	
SRP_Object_HTMLSuffix	
SRP_Object_Frame	Default is 1 Maximum is 2 (single / double frame)
SRP_Object_FrameOffset	
SRP_Object_FrameThickness	

Properties

SuperReport Pro Table Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Table_FrameColor	✔	✔	✔	color				Color of the frame See Working with colors
SRP_Table_HGridThickness	✔	✔	✔	real	1	0	10	Width of gridline horizontal borders in points
SRP_Table_VGridThickness	✔	✔	✔	real	1	0	10	Width of gridline vertical borders in points
SRP_Table_RowHeight	✔	✔	✔	real				Height of rows 0 = calculate from the style used
SRP_Table_ VariableRowHeight	✔	✔	✔	bool	yes			Rows can have variable height SRP_Table_RowHeight is used as minimum row height in this case At least one column must have SRP_Column_CalcRowHeight = 1
SRP_Table_NumColumns	✔	✔	✔	int				Number of columns in the table
SRP_Table_NumHeadings	✔			int				Number of table header rows
SRP_Table_NumFooters	✔			int				Number of table footer rows
SRP_Table_DrawHeaders	✔	✔	✔	bool	1	0	1	Print the table headers
SRP_Table_DrawColumns	✔	✔	✔	int	1	0	2	0 = no 1 = yes 2 = draw nothing (no headers) if there are no data
SRP_Table_DrawFooters	✔	✔	✔	bool	0	0	1	Print the table footers The footer is only printed when SRP_Table_ DrawColumns is True (1) and data exists; otherwise only the header is drawn (if SRP_Table_DrawColumns # 2) Footer cells are evaluated at the end of table printing (variable substitution)
SRP_Table_FixedSize	✔	✔	✔	bool	no			Use the fixed size as defined in the report
SRP_Table_AltRowOptions	✔	✔	✔	int	0	0	7	Alternate row coloring options: bit 0: 1 = enable, 0 = disable bit 1: 1 = apply SRP_Table_AltRowColor to even rows, 0 = apply to odd rows bit 2: 1 = alt color applies to empty space below the last row (if any) – usable with fixed size able only
SRP_Table_AltRowColor	✔	✔	✔	color	#FFEEEEEE			Alternate row color (default is light gray)

Table Header Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Header
SRP_Object_XML	Only when visible (not covered by another cell)
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	Read only
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	Read only
SRP_Object_PosTop	Read only
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	Not persistent, no XML name
SRP_Object_PosHeight	Not persistent, no XML name
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_Fill	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation

- SRP_Style_BaseLineShift
- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Table Header Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Header_Dynamic	✔	✔	✔	bool	no			Parse for variable substitution <%variable%> The variable substitution is performed on first data row
SRP_Header_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Header_Width	✔	✔	✔	real				Width of the header cell 0 = automatic
SRP_Header_Height	✔	✔	✔	real				Height of the header cell 0 = automatic
SRP_Header_ColSpan	✔	✔	✔	int	1	0	100	Number of columns to span over
SRP_Header_RowSpan	✔	✔	✔	int	1	0	100	Number of header rows to span over
SRP_Header_Data	✔	✔	✔	text				The header text

Table Column Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Column
SRP_Object_XML	
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	Read only
SRP_Object_PosTop	Read only
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	Not persistent, no XML name
SRP_Object_PosHeight	Not persistent, no XML name
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_Fill	
SRP_Object_Script	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation

- SRP_Style_BaseLineShift
- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Table Column Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Column_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Column_Width	✔	✔	✔	real				Width of the data column 0 = automatic
SRP_Column_Grid	✔	✔	✔	bool	yes			Draw vertical grid for this column
SRP_Column_Source	✔	✔	✔	text				Source of the data: variable name or [Table]Field
SRP_Column_Alias	✔	✔	✔	text				Alias of the column, as defined by the developer using the ***SR Variables*** command in the old SuperReport Pro API Only used in the editor
SRP_Column_Format	✔	✔	✔	text				Picture columns use formats as defined for picture objects, e.g. "3" for scaled proportionally For date/time columns, use 4D specific formats as text, e.g. "107" - can use **String** (Blank if null date + Internal date short)
SRP_Column_RowNum	✔	✔	✔	bool	no			Print row number - set when SRP_Column_Source is set to "%ROWNUM%"
SRP_Column_Duplicates	✔	✔	✔	bool	yes			Print repeated values
SRP_Column_ExecutionLevel	✔	✔	✔	int	0	0		Defines the order of execution of the scripts for columns This enables you to have 1 → M → M related data
SRP_Column_CalcRowHeight	✔	✔	✔	bool	no			Calculate row height on this column

Table Footer Properties

Common Object Properties used

Constant	Specifics
SRP_Object_Kind	Object kind = SRP_ObjectKind_Footer
SRP_Object_XML	Only when visible (not covered by another cell)
SRP_Object_ID	
SRP_Object_Name	
SRP_Object_Order	
SRP_Object_Rect	
SRP_Object_Visible	Read only
SRP_Object_Locked	
SRP_Object_Selected	
SRP_Object_PosLeft	Read only
SRP_Object_PosTop	Read only
SRP_Object_PosRight	
SRP_Object_PosBottom	
SRP_Object_PosWidth	Not persistent, no XML name
SRP_Object_PosHeight	Not persistent, no XML name
SRP_Object_RelPosRight	
SRP_Object_RelPosBottom	
SRP_Object_Fill	

Style Properties used

- SRP_Style_Features
- SRP_Style_FontName
- SRP_Style_Size
- SRP_Style_Full
- SRP_Style_Bold
- SRP_Style_Italic
- SRP_Style_Underline
- SRP_Style_StrikeThrough
- SRP_Style_Wrap
- SRP_Style_HorAlign
- SRP_Style_VertAlign
- SRP_Style_TextColor
- SRP_Style_BackColor
- SRP_Style_FrameColor
- SRP_Style_Rotation
- SRP_Style_BaseLineShift

- SRP_Style_HorizontalScale
- SRP_Style_LineSpacing
- SRP_Style_FirstLineIndent

Properties

SuperReport Pro Table Footer Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_Footer_Dynamic	✔	✔	✔	bool	no			Parse for variable substitution <%variable%>
SRP_Footer_Attributed	✔	✔	✔	bool	no			Multistyled text For more information, see the Text Style Tags section
SRP_Footer_Width	✔	✔	✔	real				Width of the footer cell 0 = automatic
SRP_Footer_Height	✔	✔	✔	real				Height of the footer cell 0 = automatic
SRP_Footer_ColSpan	✔	✔	✔	int	1	0	100	Number of columns to span over
SRP_Footer_RowSpan	✔	✔	✔	int	1	0	100	Number of footer rows to span over
SRP_Footer_Data	✔	✔	✔	text				The footer text

Get Objects

The following properties are used to define the object selection filter (selector) when calling the SR_GetObjects command to retrieve the Object numbers.

Properties

SuperReport Pro Get Objects Properties								
Constant	**Get**	**Set**	**Per**	**Type**	**Default**	**Min**	**Max**	**Comments**
SRP_ReportAllObjects	✔			int				All objects in a report
SRP_ReportSelectedObjects	✔			int				Selected objects in a report
SRP_ReportEditorStyles	✔			int				Styles used by the editor
SRP_ReportDataSource	✔			int				The data source
SRP_ReportGuides	✔			int				All guides
SRP_ReportStyleSet	✔			int				All styles
SRP_ReportSections	✔			int				All sections
SRP_SectionObjects	✔			int				All root objects in a section
SRP_GroupObjects	✔			int				All root objects in a group
SRP_TableHeaderRowMask	✔			int				All headers in a table header row Use **String** ($row;SRP_TableHeaderRowMask) to access the row number $row Numbered from 1, e.g. "H002" is the second row in headers
SRP_TableColumns	✔			int				All columns in a table
SRP_TableFooterRowMask	✔			int				All footers in a table footer row Use **String** ($row;SRP_TableFooterRowMask) to access the row number $row Numbered from 1, e.g. "F002" is the second row in footers

7 Print flags

Here is a summary of the flags (option bits) pertaining to printing SuperReport Pro areas to a printer or file and / or previewing them using PDF (Preview) on MacOS and either PDF or XPS on Windows.

Constants

The following four commands include a **DstFlags** parameter where many option bits (a.k.a. “Print flags”) can be included for settings (except SRP_Print_WinPDFPreview, only used for Preview):

- SR_Print and SR_PrintBLOB print SuperReport Pro reports
- SR_OpenSession and SR_OpenSessionBLOB create a printing session (which can be a PDF file creation and not actual printing)

In addition, the SR_PrintSettings command uses some of the Print flags in the **Options** parameter, indicated by a * in the list below.

Use the **or** operator **|** to combine them.

- SRP_Print_4DJobSetup*: use 4D job settings (**PRINT SETTINGS**)
- SRP_Print_4DPageSetup*: use 4D page settings (**PRINT SETTINGS**)
- SRP_Print_AskJobSetup*: display the system dialog for the user to define the job setup
- SRP_Print_AskPageSetup*: display the system dialog for the user to define the page setup
- SRP_Print_Default: same as SRP_Print_DestinationPrinter | SRP_Print_AskPageSetup | SRP_Print_AskJobSetup
- SRP_Print_DefaultJobSetup*: use the default printer job settings
- SRP_Print_DefaultPageSetup*: use the default printer page settings
- SRP_Print_DestinationFile: create a PDF file on MacOS and either PDF or XPS on Windows (or other, depending on the printer and driver used e.g. HP LaserJet printer could create a PostScript or PCL file)
- SRP_Print_DestinationPDF: on MacOS, equal to SRP_Print_DestinationFile
- SRP_Print_DestinationPreview: create a PDF file on MacOS and either PDF or XPS on Windows (or other, see above) then open it - see Preview
- SRP_Print_DestinationPrinter: the default job name is the report name set with SRP_Object_Name (**Edit > Report Options > Name** in user mode) - if the name is empty, “SuperReport Pro” is used
- SRP_Print_MacUseDestination: on MacOS, use the destination saved in the report (SRP_Report_PrintSettings)
- SRP_Print_NoDefaultPrinter: use the printer stored in the report, if any
- SRP_Print_NoProgress: remove the progress indicator when printing (e.g. on the server) - also used when printing to a file - can also be used as a Preview flag
- SRP_Print_SimplePageSetup*: use only report size, orientation and scaling stored in the report, not platform-native page setup / job setup)

- SRP_Print_ValidatePageSetup*: ask the user first, then display the system dialog to define the page setup if the report page setup differs from the currently used printer
- SRP_Print_WinOpenCreatedFile: on Windows, can be used in conjunction with SRP_Print_DestinationPDF or SRP_PrintDestinationFile to open the generated file (implicit in SRP_Print_DestinationPreview mode - the file will be opened automatically after generating the preview)
- SRP_Print_WinPDFNoFonts: only useful when using SuperReport Pro's internal PDF library on Windows - can also be used as a Preview flag - see also Printing to PDF on Windows
- SRP_Print_WinPDFPreview: this flag in only useful as a Preview flag and cannot be used in the **DstFlags** parameter

Print settings

- to apply the user's setting (setup in the SuperReport Pro editor or using SR_PrintSettings), specify the following Print flags: SRP_Print_NoDefaultPrinter or even SRP_Print_NoDefaultPrinter | SRP_Print_MacUseDestination
- to use 4D's settings (using **PRINT SETTINGS**): SRP_Print_4DPageSetup | SRP_Print_4DJobSetup
- to use the default printer settings: SRP_Print_DefaultPageSetup | SRP_Print_DefaultJobSetup

These options are mutually exclusive: either use what is saved in the report or 4D's settings or default printer settings.

- if you specify SRP_Print_NoDefaultPrinter, the printer name stored in the report will be used (the printer name is stored in the report **per platform**)
- if you don't specify SRP_Print_NoDefaultPrinter in the Print flags, the stored printer name will **not** be used: the default printer will be used instead.
- if you specify SRP_Print_MacUseDestination, even the destination saved in the report will be used on MacOS (if user clicked Preview, it will do a preview!)
- if you specify SRP_Print_SimplePageSetup, only the page size, orientation and scaling saved in the report will be used with default printer's settings (recommended for cross-platform printing)
- if you specify SRP_Print_ValidatePageSetup, the page setup system dialog is displayed if the report page setup differs from the currently used printer - “differs” mainly covers the paper size, margins and scaling; “currently used” depends on SRP_Print_NoDefaultPrinter, SRP_Print_4DPageSetup, SRP_Print_4DJobSetup, printer stored in the report, default printer, printer name provided in the command

Preview

The SRP_Print_DestinationPreview Print flag will create a preview file, then open it.

Preview can also be performed using the **File** menu (**File > Preview**) or the old ***SR Preview*** command.

Note: the old ***SR Preview*** command maps to SR_PrintBLOB, you can use ***SR_PrintBLOB*** directly, including the file name to use.

Preview always means “Open the file after creation”: opening the file is implicit, there is no need to explicitly set the SRP_Print_WinOpenCreatedFile Print flag.

On Windows, it will create a XPS file “SRP_Preview.xps” in User's Documents if there is no report name (i.e. if SRP_Area_WinPreviewName is empty) and no name was provided in **DstPath**.

If the SRP_Print_WinPDFPreview Preview flag is used (see below) or Microsoft XPS Document Writer is not available, a PDF will be created instead of the XPS.

Note: when using **File > Preview**, set SRP_Area_WinPreviewName if you want to change the name of the Preview file on Windows.

Preview flags

Preview flags are a subset of Print flags, useful in preview mode. These settings will affect all future previews from the Editor (**File > Preview**) or using the old ***SR Preview*** command, both cases where **DstFlags** can't be provided by any other means.

In other words, when you (as a developer) have the control, nothing is added by SuperReport Pro. When there is no other way (e.g. using the editor user interface), SRP_Area_PreviewFlags is used to specify additional flags.

Note: this property can only be set globally: in other words, it expects the value zero as the first two parameters to the getter / setter command.

These flags are set through this SRP_Area_PreviewFlags property value as the sum of one or more of the following bit constants: SRP_Print_WinPDFNoFonts, SRP_Print_WinPDFPreview, SRP_Print_NoProgress, e.g.:

SR_SetLongProperty (0;0;SRP_Area_PreviewFlags; SRP_Print_WinPDFPreview | SRP_Print_WinPDFNoFonts)

- SRP_Print_WinPDFNoFonts: only useful when using internal PDF library on Windows, do not embed fonts in the PDF file - see Printing to PDF on Windows
- SRP_Print_WinPDFPreview: force PDF preview instead of XPS even if XPS is available (only a Preview flag, not to be used in the **DstFlags** parameter of the commands)

Note: when SRP_Print_DestinationPreview | SRP_Print_WinPDFPreview is used, it is internally mapped to SRP_Print_DestinationPDF | SRP_Print_WinOpenCreatedFile.

- SRP_Print_NoProgress: remove the progress indicator when building the preview

Note: there is also a specific SRP_Export_NoProgress flag for SRP_Report_ExportFlags to suppress the progress bar while exporting.

Examples

The examples below illustrate the SRP_Print_NoProgress Print flag, used as such for printing or as a Preview flag.

Print

The following line will print the report stored in $path on the device called “my Printer” without progress bar:

```
$error:=SR_Print ($path; 0; SRP_Print_NoProgress; ""; 0; "my Printer")  // print without progress
```

Preview

The following line will set all subsequent previews triggered by the **File > Preview** menu or the old ***SR Preview*** command to remove progress bar display:

```
SR_SetLongProperty (0;0;SRP_Area_PreviewFlags; SRP_Print_NoProgress)
```

Programming a preview

If you want to programmatically display a preview without progress indicator:

```
$error:=SR_Print ($path; 0; SRP_Print_NoProgress | SRP_Print_DestinationPreview; ""; 0; "")
```

Editor and command mapping

- When preview in the Editor (**File > Preview**) is used, ***SR_Print*** is called with:
- SRP_Print_DestinationPreview | SRP_Print_NoDefaultPrinter | (value of SRP_Area_PreviewFlags)
- When the old ***SR Preview*** command is used, ***SR_Print*** is called with:

  SRP_Print_DestinationPreview | (value of the SRP_Area_PreviewFlags)

File name

The report name (or the name provided in **DstPath**) will be used for the preview file name.

- on Windows, if **DstPath** is empty the name will be “SRP_Preview.xps”

  The suffix is changed to “.pdf” if SRP_Print_WinPDFPreview is specified.

  The suffix is changed to “.xps” if SRP_Print_WinPDFPreview is not specified and the suffix is neither “.xps” nor “.oxps”.
- on MacOS, if **DstPath** is empty (which is always the case when using **File > Preview**), the printing job name is used i.e. the report name (SRP_Object_Name of the report) or "SuperReport Pro" if the report name is empty

  ```
  $error:=SR_GetPtrProperty ($areaRepRef; 1; SRP_Object_Name;->$objName)
  ```

or simpler:

```
$reportName:=SR_GetTextProperty ($areaRepRef; 1; SRP_Object_Name)
```

If the destination name is not a fully qualified name, the user's Documents directory is used on Windows and the invisible /var/folders/ system directory is used on MacOS.

Unique name on Windows

If you want to preview several reports at once, you must use a unique file name. Set the file name just before calling the print command, then reset it at the end (either to an empty string or any preferred name).

The easiest is probably to use a sequential counter and a temporary directory for the preview file to be stored. You can try to delete that directory's content on your application startup (or termination):

```
$path:=<>myTemporaryDirectory+"MyGreatestApplication Preview " + String (<>lPreview)+ ".xps"
<>lPreview:=<>lPreview+1  //increment the counter
```

Use $path directly if you are calling ***SR_PrintBLOB***.

If you are calling the old ***SR Preview*** command, do this just before the call:

```
SR_SetTextProperty (0;0;SRP_Area_WinPreviewName;$path)
```

Then reset it at the end of the loop:

```
SR_SetTextProperty (0;0;SRP_Area_WinPreviewName;"")
```

Alternatively:

- create a printing session for the preview (using ***SR_OpenSession*** with SRP_Print_DestinationPreview)
- print all reports using ***SR_PrintBLOB*** within that session
- close the session (using SR_CloseSession)

Then you only need one file.

Windows File format

The printer used for preview is Microsoft XPS Document Writer.

If no printer named “Microsoft XPS Document Writer” can be found or SRP_Print_WinPDFPreview is specified, the SuperReport Pro internal PDF library is used to create a PDF preview.

Using Microsoft XPS Document Writer on Windows

You can check **PRINTER LIST** to find out if it is present then install it if needed.

On Windows XP SP3, Microsoft XPS Document Writer is present, but Internet Explorer is used instead of XPS Viewer.

On Windows 8, you can try to use OXPS. Set the name of the preview file to be used including the extension:

```
SR_SetTextProperty (0;0;SRP_Area_WinPreviewName;"My SuperReport Pro Preview.oxps")
```

Note: SuperReport Pro always uses the same file. If you open this file in XPS Viewer, SuperReport Pro can't write to it and you get Access denied = error 5.

You must first close the preview file before SuperReport Pro can write a new preview into it.

Printing to PDF on Windows

PDF File size

You can control the size of printed reports when using the PDF destination for output on Windows when you ask SuperReport Pro not to embed fonts (SRP_Print_WinPDFNoFonts).

All used fonts are normally embedded into the PDF file, e.g. using two fonts (Arial and Times New Roman) in a small two-pages report:

- 254 Kb without the fonts
- 2.4 Mb with the fonts

Note: graphics are not downscaled (e.g. printing a 1200 dpi picture includes the picture as-is, it is not downscaled to 300 or 600 dpi) - you need PDFCreator for this.

Using PDFCreator

SuperReport Pro v3 supports PDFCreator directly: "Print to file" using PDFCreator produces a PostScript (PS) file, then SuperReport Pro calls PDFCreator to convert it to a PDF file.

The printer has to be named "PDFCreator". You can check **PRINTER LIST** to find out if it is present then install it if needed.

Here is an example of printing to a PDF using PDFCreator:

```
$error:=SR_Print ($areaRepRef;0; SRP_Print_DestinationFile;$path;0;"PDFCreator")
```

See other examples of usage with SR_SetLongProperty.

Working with Colors

You can use colors in your SuperReport Pro reports in various ways: to color text, backgrounds, or other objects.

Specifying Colors

Internally, all colors in SuperReport Pro version 3 use ARGB (alpha-red-green-blue, each channel using 8 bits: 0-255 / 0x00-0xFF).

You can use the alpha channel to specify transparency. The value should be between 0 - 255 (0x00 - 0xFF).Transparency of 0 means fully transparent (invisible) color; transparency of 255 (0xFF) means fully opaque color.

However, there are seven ways that you can specify colors in SuperReport Pro. Where necessary, they will be converted to the ARGB model. The seven methods can be split into two groups: color values passed as string values, and color values passed as longint values.

Color values passed as string values

1. Using one of the standard color names (red, green, blue, dark red, dark blue, white, gray, light gray, cyan, magenta, yellow, brown, orange, dark orange, purple, black). In this case, you pass the color name using ***SR_SetTextProperty***.

For all above values, the alpha is always 100%. You can also use “transparent”, which will set the alpha channel to 0%.

2. Using standard hexadecimal notion with one of the text commands.

e.g. “0xFFFF0000” is 100% red

3. Using hexadecimal ARGB (alpha-red-green-blue) notation. In this format, a leading # is used, followed by two hexa numbers per channel; if less than four channels are specified, full alpha (0xFF) is assumed. Note that this is the format used internally by SuperReport Pro.

e.g. “#FF0000” is the same as “#FFFF0000” = 100% red

4. 3- or 4-part RGBA comma-separated real type channel values can be used with one of the text commands. Channel values have to be in range 0.0 - 1.0; if three values are specified, alpha is assumed to be 1.0. This is simply the percentage for each color (and alpha for transparency). Note that in this case alpha is at the end. This format conversion is triggered by any “.” in the value.

e.g. “1.0,0,0” is the same as “1.0,0,0,1.0” = 100% red

5. 3- or 4-part RGBA comma-separated long integer type channel values can also be used with one of the text commands. Channel values have to be in the range 0 - 65535; if three values are specified, alpha is assumed to be 65535. Note that in this case alpha is at the end.

e.g. “65535,0,0” is the same as “65535,0,0,65535” = 100% red

6. Using the “good old” 4D 256 color palette. Any 4D 256 color palette can be specified as “Pxxx” where xxx is the palette index in range 1 – 256. For example, the following variable/field script will set the object's background color to yellow:

SR_SetTextProperty (SRArea;SRObjectPrintRef;SRP_Style_BackColor;"P2")

Color passed in longint values

7. Using a long integer with SR_SetLongProperty. In this case, nothing is assumed about the alpha channel and the alpha value needs to be specified. The color can be conveniently written in hexa notation like 0xAARRGGBB; for example 0xFF00FF00 is 100% green. However, this number in decimal notation is -16711936.

Note that the color picker and 4D RGB commands use longint values for color without the alpha channel. This means that the developer must add alpha channel information to the color if he is going to pass a color to SuperReport Pro by code - for example:

```
$SRPColor:=$Color | 0xFF000000
```

Converting RGB values

SuperReport Pro colors are very close to the format used by 4D.

In 4D, RGB colors are long integers interpreted as 0x00RRGGBB, so there are 3 channels each in range 0 - 255.

SuperReport Pro uses ARGB - 0xAARRGGBB - 4 channels each in range 0 - 255.

For example, let's examine the following RGB values from 4D:

- $red:=244
- $green:=248
- $blue:=255

Let's combine them using simple math:

```
$argb:=0xFF000000 | ($red << 16) | ($green << 8) | $blue
```

We get 0xFFF4F8FF.

Generally, to create ARGB color for use with SuperReport Pro, use

```
$argb:=($alpha << 24) | ($red << 16) | ($green << 8) | $blue
```

which is the same as

```
$argb:=($alpha * 256 * 256 * 256) + ($red * 256 * 256) + ($green * 256) + $blue
```

To create RGB color for use with 4D, use

```
$rgb:=($red << 16) | ($green << 8) | $blue
```

which is the same as

```
$rgb:=($red * 256 * 256) + ($green * 256) + $blue
```

Patterns

Patterns are no longer supported. They are interpreted by SuperReport Pro version 3 as transparency ratios (alpha channel value):

- "black" or 1: 100% (0xFF)
- "darkgray" or 4: 75% (0xC0)
- "gray" or 2: 50% (0x80)
- "lightgray" or 3: 25% (0x40)
- "white" or 0 or "none" or "" or anything else: 0% = no drawing (0x00)

Appendixes

Appendix 1: Troubleshooting

Localised Formats

When the database is set to use ',' and '.' as placeholders, you must not use localized formats (in other words always use "###,##0.00"), otherwise you must use localized formats - e.g. "### ##0,00" on some European systems.

Debugger window

If an error is encountered whilst a SuperReport Pro command is executing, and SRP_Area_TraceOnError bit 0 (trace on error in interpreted) has been set to True (default setting), the 4D debugger window will open in interpreted mode.

To turn Trace on error on:

```
SR_SetLongProperty ($areaRepRef;0;SRP_Area_TraceOnError;1)  // no alert in compiled (default)
SR_SetLongProperty ($areaRepRef;0;SRP_Area_TraceOnError;3)  // alert in compiled as well
```

To turn it off for both modes:

```
SR_SetLongProperty ($areaRepRef;0;SRP_Area_TraceOnError;0)
```

Note: this is only relevant for commands that do not return an error code.

Missing strings on 4D v11

When the form is loaded, you may be missing some language settings / resources / XLF and see ":15002", etc.

This is a known 4D v11 bug. Languages are loaded from Resources/en.lproj/SRP.xlf (for an English version of 4D), but 4D v11 does not support XLIFF files in plugins.

When you copy Resources/en.lproj/SRP.xlf into the corresponding directory of your database, you will see the strings in the editor, but in the dialogs they will still appear as :15002,xxx.

It seems that for 4D v11 it would be needed to copy all the strings from SRP.xlf into Resources/SRP.rsrc.

We can also provide a 4D method to do this. Feel free to contact our support.

Object visibility

SRP_Object_Visible is used only in the editor - to show / hide objects (e.g. you hide the second header to see a page layout for the first page). During printing this property does not exist.

Boolean properties can be set using all variants of ***SR_SetXXXProperty***.

SRP_Object_Draw can also be used. Put the following script into the variable's script (here named vText):

```
If (vText="")
   SR_SetLongProperty (SRArea;SRObjectPrintRef;SRP_Object_Draw;0)
End if
```

or simply (and better):

```
SR_SetLongProperty (SRArea;SRObjectPrintRef;SRP_Object_Draw;Num(vText#""))
```

The second one is better because when you use **View > Object Content > Value**, the property will be modified in the edited report.

Note: empty objects are not drawn (neither the background nor frame), which is compatible with SuperReport Pro version 2.x.

Forcing empty space at the top of the page

Some specific situations require that an empty space is left at the top of a page and that printing occurs only below this blank part.

However, when a new page is started, SuperReport Pro removes all empty space between objects on previous and current page. Thus the objects are positioned at the top of the page (below the top margin or header).

To force the empty space at the top of e.g. page 2:

- create an object, e.g. horizontal line, at the top of the second page (it can be even a few inches below the top of the second page)
- set it to stick at that position (**Fix Vertical Position** - then objects above it can move using variable depth printing and this line will not be moved into the first page) and make it invisible by changing the line color (set alpha to 0 to be fully transparent)

Now when the second page starts, that line will be positioned on top of the page. All objects below will start at the expected position (relative to this line).

Appendix 2: Hints and Tips

Quotes and Double Quotes

When writing XML code, SuperReport Pro supports the use of single quotes instead of double quotes.

For example you can use *font='Times New Roman'* - you don't have to add lots of '+**Char**(Double Quote)' to your code.

Setting the font attributes

To set the fonts for variables and fields:

SR_SetTextProperty ($areaRepRef;$objNum;SRP_Style_FontName;$font)

SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Size;$fontSize)

SR_SetLongProperty ($areaRepRef;$objNum;SRP_Style_Full;$fontStyle)

// this line modifies SRP_Style_Bold, SRP_Style_Underline, SRP_Style_Italic and SRP_Style_StrikeThrough

To set the styles:

SR_SetLongProperty ($areaRepRef;$objNum;SRP_Object_StyleID;$value)

// style to be used, all overridden properties are cleared (e.g. font name)

Modifying the font size during printing

It is possible to write a script that variably controls the font display size, based on values stored into a database record or from any other source.

For example, having a Variable / Field object on the page, make it to print variable height and create a script:

SR_SetLongProperty (SRArea; SRObjectPrintRef; SRP_Style_Size; $fontSize)

This is equivalent to the old API (see ***SR_SetFont*** in the SuperReport Pro 2.9 manual):

$error:=***SR Set Object Format*** (SRArea; SRObjectPrintRef; SR Attribute Font Size; ""; $fontSize)

Components

All 4D variables are created / accessed in the current scope.

Therefore if you are using SuperReport Pro in a component, it will access the component's variables.

Platform native settings and printing destination

The following properties hold the platform native settings:

- SRP_Report_PageFormat - Page Setup on MacOS - flattened PMPageFormat (XML)

  contains all the paper handling options including selected printer

- SRP_Report_PrintSettings - Job Setup on MacOS - flattened PMPrintSettings (XML)

  contains selected destination (including file name), number of copies, tray handling…

- SRP_Report_MacPrinter - device name from the page setup XML stored in the SRP_Report_PageFormat property

  get the device name

- SRP_Report_DevMode - binary DEVMODE structure on Windows

  holds all the page setup and job handling options

- SRP_Report_DeviceNames - binary DEVNAMES structure on Windows

  holds the selected printer

- SRP_Report_WinPrinter - device name from the DEVNAMES structure stored in the SRP_Report_DeviceNames property

  get the device name

Other properties related to printing (no constants, use the specified strings instead):

"pgsc" - old MacOS 7 Page Setup when an old SuperReport Pro report is converted.
This is only used to initialize platform native page setup when none already exists (which then should be used), this property is never modified by SuperReport Pro v3.

"land" - landscape orientation

"prsc" - page scale, cleared when ***SR_PrintSettings*** is used (this includes **File>Page Setup**, **File>Job Setup** and old commands ***SR Page Setup*** & ***SR Validate***)

Note: "land" and "prsc" are set when a page setup (on any platform) is used. They are not to be manipulated directly, unless SRP_Print_SimplePageSetup is to be used (see below). When you change them, you have to modify the margins, page size, etc.

When SRP_Print_SimplePageSetup is specified, platform native settings are ignored, only landscape and scaling are used (and paper size, of course).

When SRP_Print_NoDefaultPrinter is specified (and printer is not explicitly named), the stored printer name is used.

When SRP_Print_MacUseDestination is specified, the destination used is the one that has been stored in SRP_Report_PrintSettings.

Embedding variables in HTML export text

When a text is parsed for variables, the leading **+** sign means “don't HTML encode the value of the variable” otherwise the value is HTML encoded, but only if the text is attributed.

Note: if the text is not attributed, the contents of the variables is not HTML encoded.

Exporting static text into HTML:

- If “export plain text” is specified and the text is attributed, remove the tags, leave plain text, mark as not attributed.
- If HTML prefix or HTML suffix are not empty:
  1. Write the HTML prefix (not HTML encoded) while parsing for variable substitutions (no HTML encoding is performed)
  2. If attributed, write the text (not HTML encoded) otherwise write the HTML encoded text
  3. Write the HTML suffix while parsing for variable substitutions (no HTML encoding is performed)
- If both HTML prefix and HTML suffix are empty:
  1. Write "" (with name and id, if not empty)
  2. If attributed, write the text (not HTML encoded) otherwise write the HTML encoded text
  3. Write "",

In other words, when “export plain text” option is not specified, if you want to modify the following variables style, you must specify a HTML prefix or a HTML suffix (single space will do), not to enclose e.g. "<h1>" into "".

- not attributed text: "<h1>" or "<%vH1%>" with 4D variable vH1:="<h1>" will produce "<h1>" in the HTML stream
- attributed text: "<h1>" or "<%+vH1%>" with 4D variable vH1:="<h1>" will produce "<h1>" in the HTML stream
- attributed text: "<%vH1%>" with 4D variable vH1:="<h1>" will produce "<h1>" in the HTML stream

Note: this is a big difference from previous versions.

Setting multiple objects

If you need to modify a report, do it before printing: first create an offscreen report with the SR_NewReport command, manipulate it, print or save it, then destroy it.

To globally set e.g. the multistyled (attributed) property of all fields:

```
// select the document to work on
$doc:=Select document("";".srp";"Select a SuperReport Pro report";Package open)
$areaRepRef:=0
$error:=SR_NewReport ($areaRepRef)  // create an offscreen area
$error:=SR_LoadReport ($areaRepRef;Document;1)  // load the document
ARRAY LONGINT($objs;0)
$error:=SR_GetObjects ($areaRepRef;1;SRP_ReportAllObjects;$objs)  // get all objects
For ($i;1;Size of array($objs))
   If (SR_GetTextProperty ($areaRepRef;$objs{$i};SRP_Object_Kind)=SRP_ObjectKind_Field)  // is it a field?
      SR_SetLongProperty ($areaRepRef;$objs{$i};SRP_Field_Attributed;1)  // make it multistyled
   End if
End for
$error:=SR_SaveReport ($areaRepRef;Document;1)  // save the modified report
$error:=SR_DeleteReport ($areaRepRef)  // dispose the offscreen area
```

Overflow

“Overflow” is when a section is split over page boundaries:

- “No”: don't print.
- “Yes”: print once (this means it will be printed even on Overflow if this object was not yet printed).
- “On Overflow”: print only after the section was split (on continuing the section on next page).
- “Always”: always print.

“On Overflow” can be used e.g. for text object “Continued...” at top of a section – it will print only after the section is split.

Just create 4 text objects (No, Yes, On Overflow, Always) with fixed vertical position and try with a section that will be split:

Bottom of first page :

- “Yes”
- “Always”

Top of second page:

- “On Overflow”
- “Always”

Appendix 3: Frequently Asked Questions

User environment

Can I use SuperReport Pro in the User Environment?

Yes, SuperReport Pro can be used on any valid input form, either in the User Environment or a custom method which calls **MODIFY SELECTION**.

Compatibility

Does SuperReport Pro support 64-bit Windows systems?

Yes!

Does SuperReport Pro support Macintosh PowerPC systems?

The shipping plugin doesn't. However, we can provide a compatible version if needed - please contact us through www.e-node.net.

Does SuperReport Pro support drag and drop?

No.

Repeating Objects and Relationships

Can I use the Repeating Objects feature for multi-level relationships? For example, if I have a Customer, Invoice, and Line Item table, can I use repeating objects to show all the related Invoices, and show their related Line Items?

No, the Repeating Objects feature is designed to work with a single related many table.

In order to create this type of report, you are going to have to create it based on the Line Item's table, and using Break Header data, you can show the related one and its associated related one information.

You can use a Table object which supports 1 → M → M → M → M relations.

Virtual structure

Does SuperReport Pro use a virtual structure, if it has been defined, to look up 4D fields?

Yes. If a virtual structure is in effect and a field is not defined in the virtual structure, the physical field name is used.

Headers and footers

Which header and footer will be printed if the report has only one page?

This depends on the SRP_Section_FirstPage, SRP_Section_SecondPage and SRP_Section_LastPage properties:

- The first header/footer having both SRP_Section_FirstPage and SRP_Section_LastPage set to 1
- The first header/footer having one of SRP_Section_FirstPage and SRP_Section_LastPage set to 1 and SRP_Section_SecondPage set to 0

When multiple pages are to be printed, the behavior is as follows:

- For the first page, the first header/footer having SRP_Section_FirstPage set to 1
- For the last page, the first header/footer having SRP_Section_LastPage set to 1
- For other pages, the first header/footer having SRP_Section_SecondPage set to 1

What is the difference between setting the headers/footers from the View Menu and the checkboxes in the properties of the Footer Section?

Checkboxes in section properties for first/second/last page are used to make the choice.

View > Headers/Footers For > XXX makes use of them in the editor.

For example if there is a section to be shown on first page only and you choose to show sections for the last page, it will be made invisible.

This view is mostly useful when you have three different headers (one for first, one for last and one for all other pages)…

I have set up a report with a Header to be printed every page, A first page header to be printed after the Everypage header but only on the first page then a sub header to be printed every page followed by the body.

The First page only header does not print… ever.

When I use the menu **View > Headers/Footers for > First page** the section, which is supposed to print first page only is not showing. In fact that section does not appear at any time except when the option selected is all pages. All sections are selected for printing from the Database Menu

You can have only one header for the first page.

If there are more headers with “Print on first page”, the first one will be used.

The same holds for other page options - first one having “second page”/“last page” will be used.

And the same holds for Footers.

Thus modify the first header to contain all information for the first page, make it print on first page only.

Modify the second header to print on second and last pages and to contain all information for all pages except the first one.

Another possibility: use header for all pages, make a new sub-header (**Edit>Add Section**: break header, level 1) for the other information and make it to print on change of a constant variable (e.g. SRDate) - it will be printed only on the first page.

Plug-in Area type

It is indicated in the manual that the Plug-in Area type should be “SuperReport Pro” and not “SuperReport”.
In the example database, the type is “SuperReport”. We have not seen any difference. Which is the correct setting?

Both behave identically. Prefer ***%SuperReport*** in case changes will be needed in the future to enhance support User mode.

In version prior to 2.0 only ***%SuperReport*** was available, but is not visible in the User environment.
Then ***_SuperReport Pro*** was added in version 2.x to be able to use it in User mode.

Menus

Is it possible to replace one of the menu items with my own code?

Yes - use ***SR Menu Item***

Parameters text, enable, mark and overrideMethod are input (depending on action) and output (always).

For example, to associate a method with **File>Export**:

```
$error:=SR Menu Item ($areaRepRef; SR MenuItem Set 4D Method; SR MenuItem Print To Disk; "";0; 0; \
    "methodNameToCall")
```

For new menu items (without a new constant), consult the SRProMenus.XML file in the bundle.

When the SRP_Area_CallbackVersion property is set to 1:

- The menu event callback will expect a return value when called with SR Menu Item Selected Before: add **C_LONGINT** ($0) and set it to 1 if SuperReport Pro should handle the menu item
- The menu item callback will be provided with 2 arguments (area and menu ID): add **C_LONGINT** ($1; $2)

Is it possible to intercept SuperReport Pro menu calls via a callback method and let or not SuperReport Pro execute the appropriate command depending on the programmer's choice ($0 of the callback method)?

Yes, but only when using a menu callback method (SRP_Report_CallbackMethEvent) and SRP_Area_CallbackVersion = 1.

- The user chooses a menu item
- If a menu event callback is defined, it is called with SR Menu Item Selected Before and menu ID
- If a command is set for the menu item, it is executed, otherwise the standard action is taken
- If a menu event callback is defined, it is called with SR Menu Item Selected After and the menu ID

To disable standard execution, make the respective menu item disabled…

The menu item method gets no parameters (for compatibility), but to provide it with areaID and menuID is possible (can be made optional - call with arguments or not).

See also the discussion above regarding the SRP_Area_CallbackVersion property.

Watermarks

Can I add a watermark to my report?

Yes, you can add a Watermark section (see the User Guide for information about adding a Watermark Section, and the Watermark Properties section for information about procedurally controlling watermarks).

Multiple Undos

Is there a limit to the undo buffer?

Yes - it is 65536, which is virtually unlimited.

SRArea variable and Area/Report reference during printing

According to the manual the variable SRArea (printed report reference) can be used in callback scripts. This variable is set to -1.

The value -1 is OK. Just try to use it. Note that when there are more reports printing (e.g. on the server) they will use -2, -3…

Reports in SuperReport Pro areas and offscreen reports use positive numbers, reports being printed use negative values.

Note that the SRArea variable is only valid during printing.

If I pass in this value to ***SR Get Object IDs*** I get an error -8008 Invalid area.

Only ***SR_GetXXXProperty*** / ***SR_SetXXXProperty*** were supported during printing before version 3.3.

As of v3.3, SR_GetObjects, SR_GetObjectsByPropertyValue and old ***SR Get Object IDs*** are supported during printing. However:

- ***SR_GetObjects*** ignores the second parameter and only SRP_ReportAllObjects as the selector is usable (returning all objects of the report),
- ***SR Get Object IDs*** does not support “selected objects”, only “all objects”.

What can I pass in to ***SR Get Object IDs***?

Only a valid SuperReport Pro Area - on a form, in an external window or an offscreen report.

I am trying to get the ObjectID from a specific variable name. Is there another way of doing this?

Not during printing.

Manipulate that object in its script. SRArea and SRObjectPrintRef are valid for properties manipulation..

SuperReport Pro supports only variable / field properties change during printing (font, size, color…). E.g. you can change a color of currently printed object in the object script:

SR_SetTextProperty (SRArea;SRObjectPrintRef;SRP_Style_TextColor;"red")

I tried to use ***SR_FindObjectByID*** using err:=***SR_FindObjectByID***(SRArea;"myname";myref), which returns an error 4 and myref as 0 although an object with the given ID (which is unique) does exist.

The only API usable during printing is ***SR_GetXXXProperty*** and ***SR_SetXXXProperty***.

Appendix 4: Property Values, Constants and XML Names

Property Constant	Property Value (selector)	Property XML Name
SRP_Area_CallbackMethEdit	edit	editorProc
SRP_Area_Copyright	copy	
SRP_Area_CreatedObject	creo	
SRP_Area_DoubleClick	evtD	
SRP_Area_DraggedObject	drgo	
SRP_Area_DraggedPosH	drpX	
SRP_Area_DraggedPosV	drpY	
SRP_Area_DrawingMode	drmo	
SRP_Area_DrawingPage	drpa	
SRP_Area_Event	evtT	
SRP_Area_EventChar	evtC	
SRP_Area_EventKey	evtK	
SRP_Area_EventModifiers	evtM	
SRP_Area_EventPosH	evtH	
SRP_Area_EventPosV	evtV	
SRP_Area_ExtensionHTML	sHTM	
SRP_Area_ExtensionSRP	sSRP	
SRP_Area_ExtensionTXT	sTXT	
SRP_Area_ExtensionXML	sXML	
SRP_Area_HitObject	hito	
SRP_Area_HitPart	hite	
SRP_Area_IsArea	isEA	
SRP_Area_LastError		
SRP_Area_Moving	move	
SRP_Area_Name	anam	
SRP_Area_NewReport	defr	
SRP_Area_NotificationCallback	palp	
SRP_Area_Path	path	
SRP_Area_PreviewFlags	sPrF	
SRP_Area_Proximity	snap	
SRP_Area_Redraw	upds	
SRP_Area_Report	rept	
SRP_Area_Rounding	rund	
SRP_Area_ScrollLeft	scrl	
SRP_Area_ScrollTop	scrt	
SRP_Area_ScrollWheel	scrw	
SRP_Area_Selected	asel	
SRP_Area_Self	self	
SRP_Area_Tool	tool	
SRP_Area_ToolTips	TIPS	
SRP_Area_TraceOnError	TRAC	

Property Constant	Property Value (selector)	Property XML Name
SRP_Area_VarArea	svar	
SRP_Area_VarBegHTML	svbh	
SRP_Area_VarCurrentRun	svcr	
SRP_Area_VarDate	svda	
SRP_Area_VarDateTime	svdt	
SRP_Area_VarEndHTML	sveh	
SRP_Area_VarName	svna	
SRP_Area_VarObject	svob	
SRP_Area_VarPage	svpa	
SRP_Area_VarPages	svp#	
SRP_Area_VarPrintSection	svps	
SRP_Area_VarRecord	svre	
SRP_Area_VarRepeat	svci	
SRP_Area_VarTime	svti	
SRP_Area_Version	vers	
SRP_Area_Visible	visi	
SRP_Area_WinPreviewName	sPrN	
SRP_Column_Alias	alis	alias
SRP_Column_Attributed	attr	attributed
SRP_Column_CalcRowHeight	varh	varHeight
SRP_Column_Duplicates	dups	duplicates
SRP_Column_ExecutionLevel	exel	level
SRP_Column_Format	fmt	format
SRP_Column_Grid	grid	grid
SRP_Column_RowNum	rown	rownum
SRP_Column_Source	src	source
SRP_Column_Width	widt	width
SRP_DataSource_Alias	alis	alias
SRP_DataSource_BodyScript	scrB	BodyScript
SRP_DataSource_BodyScriptToks	scTB	BodyScriptTokens
SRP_DataSource_Callback	call	callback
SRP_DataSource_EndScript	scrE	EndScript
SRP_DataSource_EndScriptToks	scTE	EndScriptTokens
SRP_DataSource_Iterations	iter	iterations
SRP_DataSource_RelateMany	relM	relateMany
SRP_DataSource_RelateOne	rel1	relateOne
SRP_DataSource_Source	src	source
SRP_DataSource_StartScript	scrS	StartScript
SRP_DataSource_StartScriptToks	scTS	StartScriptTokens
SRP_DataSource_TableID	tbid	tableID
SRP_DataSource_Type	type	type
SRP_DataSource_VarName	name	name
SRP_Field_Alias	alis	alias

Property Constant	Property Value (selector)	Property XML Name
SRP_Field_Attributed	attr	attributed
SRP_Field_Calculate	calt	calc
SRP_Field_DrawEmpty	drem	empty
SRP_Field_Format	fmt	format
SRP_Field_KeepTogether	keep	keepTogether
SRP_Field_ParsedData	text	
SRP_Field_PrintEmptyText	txtF	printEmptyText
SRP_Field_Record	calr	record
SRP_Field_Repeat	repe	repeat
SRP_Field_RepeatOffset	repo	repeatOffset
SRP_Field_Source	src	source
SRP_Field_UseOld	oldv	useOld
SRP_Footer_Attributed	attr	attributed
SRP_Footer_ColSpan	cspn	colspan
SRP_Footer_Data	data	text
SRP_Footer_Dynamic	dyna	dynamic
SRP_Footer_Height	high	height
SRP_Footer_RowSpan	rspn	rowspan
SRP_Footer_Width	widt	width
SRP_Group_Objects	objs	
SRP_GroupObjects	objs	
SRP_Guide_Position	data	pos
SRP_Guide_Type	type	
SRP_Header_Attributed	attr	attributed
SRP_Header_ColSpan	cspn	colspan
SRP_Header_Data	data	text
SRP_Header_Dynamic	dyna	dynamic
SRP_Header_Height	high	height
SRP_Header_RowSpan	rspn	rowspan
SRP_Header_Width	widt	width
SRP_Line_Color	lclr	lineColor
SRP_Line_Flags	flgs	flags
SRP_Line_Thickness	thic	thickness
SRP_Object_Align	algn	align
SRP_Object_BindToParentH	bndH	bindH
SRP_Object_BindToParentV	bndV	bindV
SRP_Object_DashStyle	strs	stroke
SRP_Object_Draw	draw	draw
SRP_Object_Export	expo	export
SRP_Object_Fill	fill	fill
SRP_Object_FixH	fixH	fixH
SRP_Object_FixV	fixV	fixV
SRP_Object_Frame	fram	frame

Property Constant	Property Value (selector)	Property XML Name
SRP_Object_FrameOffset	foff	frameOffset
SRP_Object_FrameThickness	fthi	frameThickness
SRP_Object_HTMLPrefix	htmP	HTMLStartTag
SRP_Object_HTMLSuffix	htmS	HTMLEndTag
SRP_Object_ID	id	id
SRP_Object_Kind	kind	
SRP_Object_Locked	lock	locked
SRP_Object_Name	name	name
SRP_Object_Order	objZ	
SRP_Object_PosBottom	posb	bottom
SRP_Object_PosHeight	posh	height
SRP_Object_PosLeft	posl	left
SRP_Object_PosRight	posr	right
SRP_Object_PosTop	post	top
SRP_Object_PosWidth	posw	width
SRP_Object_Rect	rect	
SRP_Object_RelMoveH	movH	
SRP_Object_RelMoveV	movV	
SRP_Object_RelPosBottom	rpob	
SRP_Object_RelPosLeft	rpol	
SRP_Object_RelPosRight	rpor	
SRP_Object_RelPosTop	rpot	
SRP_Object_Script	scrp	Script
SRP_Object_ScriptTokens	scrT	ScriptTokens
SRP_Object_Selected	selc	selected
SRP_Object_StyleID	styl	styl
SRP_Object_VariableSizeH	expH	varSizeH
SRP_Object_VariableSizeV	expV	varSizeV
SRP_Object_Visible	visi	visible
SRP_Object_XML	XML	
SRP_Oval_Color	lclr	lineColor
SRP_Oval_FillColor	Fclr	fillColor
SRP_Oval_Thickness	thic	thickness
SRP_Picture_Data	data	ImageData
SRP_Picture_FillColor	Fclr	fillColor
SRP_Picture_Format	fmt	format
SRP_Picture_FrameColor	fclr	frameColor
SRP_Picture_Height	high	
SRP_Picture_HTMLReplace	htmR	HTMLReplacementTag
SRP_Picture_Size	imgs	
SRP_Picture_Width	widt	
SRP_Rect_Color	lclr	lineColor
SRP_Rect_Columns	cols	cols

Property Constant	Property Value (selector)	Property XML Name
SRP_Rect_FillColor	Fclr	fillColor
SRP_Rect_Flags	flgs	flags
SRP_Rect_Rows	rows	rows
SRP_Rect_Thickness	thic	thickness
SRP_Report_BasicSRMenu	srmb	
SRP_Report_CallbackMethEvent	call	eventProc
SRP_Report_CallbackVersion	cbcv	
SRP_Report_ColumnCount	colC	columnCount
SRP_Report_ColumnOrder	colH	printOrder
SRP_Report_ColumnSpacing	colS	columnSpacing
SRP_Report_CountPages	cpgs	countPages
SRP_Report_DataSource	DATA	
SRP_Report_DeviceNames	wdnh	DeviceNames
SRP_Report_DevMode	wdmh	DevMode
SRP_Report_Document	docn	
SRP_Report_DoMenu	domi	
SRP_Report_DoMenuNoProc	domI	
SRP_Report_EditNumberPages	nump	numPages
SRP_Report_EnableScripts	enas	
SRP_Report_ExportDelimiter	expD	
SRP_Report_ExportFlags	expO	
SRP_Report_ExportRecDelimiter	expR	
SRP_Report_GridSize	grsi	gridSize
SRP_Report_HideHTML	enah	
SRP_Report_LockGuides	gulo	lockGuides
SRP_Report_LockSections	selo	lockSections
SRP_Report_MacPrinter	Mprn	
SRP_Report_MarginBottom	marB	pageMarginBottom
SRP_Report_MarginLeft	marL	pageMarginLeft
SRP_Report_MarginRight	marR	pageMarginRight
SRP_Report_Margins	marg	pageMargins
SRP_Report_MarginTop	marT	pageMarginTop
SRP_Report_Modified	modi	
SRP_Report_ObjectHierarchy	Olst	
SRP_Report_PageFormat	pgfm	PageFormat
SRP_Report_PageHeight	high	pageHeight
SRP_Report_PageWidth	widt	pageWidth
SRP_Report_PhysicalPaper	phys	usePhysical
SRP_Report_PrintEmptyText	txtF	printEmptyText
SRP_Report_PrintSettings	pgst	PrintSettings
SRP_Report_RulerUnits	rulu	rulerUnits
SRP_Report_Scale	resc	reportScale
SRP_Report_ShowGrid	grsh	showGrid

Property Constant	Property Value (selector)	Property XML Name
SRP_Report_ShowGuides	gush	showGuides
SRP_Report_ShowMargins	mars	showMargins
SRP_Report_ShowMenubar	mens	showMenubar
SRP_Report_ShowObjBorders	obor	showObjBorders
SRP_Report_ShowRuler	rush	showRulers
SRP_Report_ShowSections	sesh	showSections
SRP_Report_ShowSRMenu	srms	
SRP_Report_ShowToolbar	toos	showToolbar
SRP_Report_ShowZoom	zoms	
SRP_Report_SnapToGrid	grsn	snapToGrid
SRP_Report_SnapToGuides	gusn	snapToGuide
SRP_Report_UserBLOB	usrb	
SRP_Report_Version	vers	version
SRP_Report_VirtualCommands	Clst	
SRP_Report_VirtualStructure	Slst	
SRP_Report_VirtualVariables	Vlst	
SRP_Report_WinPrinter	Wprn	
SRP_Report_Zoom	scal	scale
SRP_Report_ZoomTitle	zoom	
SRP_ReportAllObjects	objs	
SRP_ReportDataSource	DATA	
SRP_ReportEditorStyles	STYL	
SRP_ReportGuides	GUID	
SRP_ReportSections	body	
SRP_ReportSelectedObjects	selc	
SRP_ReportStyleSet	STL#	
SRP_Section_Break_Alias	alis	alias
SRP_Section_Break_Always	alwa	always
SRP_Section_Break_Level	brkL	level
SRP_Section_Break_On	brkO	
SRP_Section_Break_OnArray	brkA	breakOnArray
SRP_Section_Break_OnField	brkF	breakOnField
SRP_Section_Break_OnVariable	brkV	breakOnVariable
SRP_Section_Break_Type	brkT	
SRP_Section_FirstPage	pag1	firstPage
SRP_Section_FixedHeight	fixd	fixedHeight
SRP_Section_FooterFromBottom	bind	bindToBottom
SRP_Section_Height	high	height
SRP_Section_KeepTogether	keep	keepTogether
SRP_Section_LastPage	pagN	lastPage
SRP_Section_MinSpace	mins	minSpace
SRP_Section_PageThrow	pthr	pageThrow
SRP_Section_SecondPage	pag2	secondPage

Property Constant	Property Value (selector)	Property XML Name
SRP_Section_Type	type	
SRP_Section_Watermark_OnTop	onto	onTop
SRP_SectionObjects	objs	
SRP_Style_BackColor	bclr	backColor
SRP_Style_BaseLineShift	basl	baseLineShift
SRP_Style_Bold	styB	bold
SRP_Style_Features	sfea	features
SRP_Style_FontName	fnam	font
SRP_Style_FrameColor	fclr	frameColor
SRP_Style_Full	styF	qdStyle
SRP_Style_HorAlign	halg	halign
SRP_Style_HorizontalScale	hors	hScale
SRP_Style_Italic	styI	italic
SRP_Style_LineSpacing	lisp	lineSpacing
SRP_Style_Rotation	rotd	rotation
SRP_Style_Size	size	size
SRP_Style_StrikeThrough	styS	strikethrough
SRP_Style_TextColor	tclr	textColor
SRP_Style_Underline	styU	underline
SRP_Style_VertAlign	valg	valign
SRP_Style_Wrap	wrap	wrap
SRP_Table_AltRowColor	altc	altRowColor
SRP_Table_AltRowOptions	alto	altRowColorParams
SRP_Table_DrawColumns	codr	draw
SRP_Table_DrawFooters	fodr	draw
SRP_Table_DrawHeaders	hedr	draw
SRP_Table_FixedSize	fixs	fixedSize
SRP_Table_FrameColor	fclr	frameColor
SRP_Table_HGridThickness	hgrt	hGridThickness
SRP_Table_NumColumns	coln	cols
SRP_Table_NumFooters	ftrn	
SRP_Table_NumHeadings	hdrn	
SRP_Table_RowHeight	high	height
SRP_Table_VariableRowHeight	varh	varHeight
SRP_Table_VGridThickness	vgrt	vGridThickness
SRP_TableColumns	colu	
SRP_TableFooterRowMask	F000	
SRP_TableHeaderRowMask	H000	
SRP_Text_Attributed	attr	attributed
SRP_Text_Data	data	text
SRP_Text_DrawEmpty	drem	empty
SRP_Text_Dynamic	dyna	dynamic
SRP_Text_KeepTogether	keep	keepTogether

Property Constant	Property Value (selector)	Property XML Name
SRP_Text_ParsedData	text	
SRP_Text_PrintEmptyText	txtF	printEmptyText
SRP_Variable_Alias	alis	alias
SRP_Variable_Attributed	attr	attributed
SRP_Variable_Calculate	calt	calc
SRP_Variable_DrawEmpty	drem	empty
SRP_Variable_Format	fmt	format
SRP_Variable_Index	elem	elem
SRP_Variable_KeepTogether	keep	keepTogether
SRP_Variable_ParsedData	text	
SRP_Variable_PrintEmptyText	txtF	printEmptyText
SRP_Variable_Record	calr	record
SRP_Variable_Repeat	repe	repeat
SRP_Variable_RepeatOffset	repo	repeatOffset
SRP_Variable_Source	src	source
SRP_Variable_UseOld	oldv	useOld

Appendix 5: Other Constants

The constants not directly referring to properties are listed belown by themes.

This Appendix contains two sections:

- New API constants (version 3 and above, prefixed by "SRP", no spaces in constant names).
- Legacy constants (from previous versions, prefixed by "SR Pro", no underscores in constant names).

Version 3 API constants

SRP Object Kinds

Constant	Type	Value
SRP_ObjectKind_Style	S	Style
SRP_ObjectKind_Area	S	SuperReport Pro
SRP_ObjectKind_Report	S	Report
SRP_ObjectKind_Section	S	Section
SRP_ObjectKind_Group	S	Group
SRP_ObjectKind_Line	S	Line
SRP_ObjectKind_Oval	S	Oval
SRP_ObjectKind_Rectangle	S	Rect
SRP_ObjectKind_Picture	S	Pict
SRP_ObjectKind_Text	S	Text
SRP_ObjectKind_Variable	S	Var
SRP_ObjectKind_Field	S	Field
SRP_ObjectKind_Table	S	Table
SRP_ObjectKind_Header	S	Header
SRP_ObjectKind_Column	S	Column
SRP_ObjectKind_Footer	S	Footer
SRP_ObjectKind_DataSource	S	DataSource
SRP_ObjectKind_Guide	S	Guide

SRP Section Types

Constant	Type	Value
SRP_SectionType_Header	S	Header
SRP_SectionType_BreakHeader	S	BreakHeader
SRP_SectionType_Body	S	Body
SRP_SectionType_BreakFooter	S	BreakFooter
SRP_SectionType_Footer	S	Footer
SRP_SectionType_Watermark	S	Watermark

SRP Create Object Types

Constant	Type	Value
SRP_Group	S	GRP#
SRP_Line	S	LINE
SRP_Oval	S	OVAL
SRP_Rectangle	S	RECT
SRP_Picture	S	PICT
SRP_Text	S	TEXT
SRP_Variable	S	VARI
SRP_Field	S	FLD
SRP_Table	S	TABL
SRP_Header	S	HDrs
SRP_BreakHeader	S	HDBR
SRP_Body	S	body
SRP_BreakFooter	S	FOBR
SRP_Footer	S	FOOs
SRP_Watermark	S	wate
SRP_Scrap	S	scrs
SRP_HorizontalGuide	S	GUHo
SRP_VerticalGuide	S	GUVe
SRP_DataSource	S	DATA
SRP_Style	S	STYL

SRP Area Tools

Constant	Type	Value
SRP_Tool_Select	L	0
SRP_Tool_Text	L	1
SRP_Tool_Field	L	2
SRP_Tool_Variable	L	3
SRP_Tool_Line	L	4
SRP_Tool_Rectangle	L	5
SRP_Tool_Oval	L	6
SRP_Tool_Picture	L	7
SRP_Tool_Table	L	8

SRP Get Objects Types

Constant	Type	Value
SRP_GroupObjects	S	objs
SRP_ReportAllObjects	S	objs
SRP_ReportDataSource	S	DATA
SRP_ReportEditorStyles	S	STYL
SRP_ReportGuides	S	GUID
SRP_ReportSections	S	body
SRP_ReportSelectedObjects	S	selc
SRP_ReportStyleSet	S	STL#
SRP_SectionObjects	S	objs
SRP_TableColumns	S	colu
SRP_TableFooterRowMask	S	F000
SRP_TableHeaderRowMask	S	H000

SRP Object Binding

Constant	Type	Value
SRP_Object_Bind_None	L	0
SRP_Object_Bind_Move	L	1
SRP_Object_Bind_Grow	L	2

SRP Style Features

Constant	Type	Value
SRP_Style_HasBaseStyle	L	1
SRP_Style_HasFontName	L	2
SRP_Style_HasFontSize	L	4
SRP_Style_HasFontStyle	L	8
SRP_Style_HasTextColor	L	16
SRP_Style_HasBackColor	L	32
SRP_Style_HasHorAlign	L	64
SRP_Style_HasVertAlign	L	128
SRP_Style_HasWrap	L	256
SRP_Style_HasFrameColor	L	512
SRP_Style_HasRotation	L	1024
SRP_Style_HasBaseLineShift	L	2048
SRP_Style_HasHorizontalScale	L	4096
SRP_Style_HasLineSpacing	L	8192
SRP_Style_HasFirstLineIndent	L	16384

SRP Error Codes

Constant	Type	Value
SRP_Err_OK	L	0
SRP_Err_Generic	L	-1
SRP_Err_CantLoadXML	L	-2
SRP_Err_CantSaveXML	L	-3
SRP_Err_InvalidAreaRef	L	-4
SRP_Err_InvalidObjectRef	L	-5
SRP_Err_InvalidRequest	L	-6
SRP_Err_InvalidArrayType	L	-7
SRP_Err_InvalidNilPointer	L	-8
SRP_Err_InvalidPointerType	L	-9
SRP_Err_InvalidArraySize	L	-10
SRP_Err_InvalidSession	L	-11
SRP_Err_UserCancelled	L	-128
SRP_Err_HdrFtrTooBig	L	-205
SRP_Err_NoFitInSectionOrGroup	L	-206
SRP_Err_NoFitInFixedSizeGroup	L	-207
SRP_Err_HdrTooBig	L	-210
SRP_Err_SectionTooBig	L	-211
SRP_Err_TooManySubPages	L	-212
SRP_Err_TooManyPages	L	-213
SRP_Err_SectionTooBigForNewPage	L	-214
Abort (button in runtime error dialog or 4D error)	L	-1000

SRP Export Flags

Constant	Type	Value
SRP_Export_NoProgress	L	8
SRP_Export_CSVFormat	L	16
SRP_Export_StaticText	L	32
SRP_Export_Sorted	L	64
SRP_Export_PlainText	L	128
SRP_Export_Text	L	256
SRP_Export_XML	L	512
SRP_Export_HTML	L	1024
SRP_Export_Body	L	4096
SRP_Export_Breaks	L	8192
SRP_Export_Total	L	16384
SRP_Export_Headers	L	32768
SRP_Export_Watermark	L	65536

SRP Print Picture

Constant	Type	Value
SRP_PrintPict_PDFVector	L	0
SRP_PrintPict_EMFVector	L	0
SRP_PrintPict_PNG	L	1
SRP_PrintPict_TIFF	L	2
SRP_PrintPict_JPEG	L	3
SRP_PrintPict_BMP	L	4
SRP_PrintPict_GIF	L	5
SRP_PrintPict_PDFBitmap	L	6

SRP Print Flags

Constant	Type	Value
SRP_Print_DestinationPrinter	L	0
SRP_Print_DestinationFile	L	1
SRP_Print_DestinationPreview	L	3
SRP_Print_DestinationPDF	L	4
SRP_Print_ValidatePageSetup	L	16
SRP_Print_DefaultPageSetup	L	32
SRP_Print_4DPageSetup	L	64
SRP_Print_SimplePageSetup	L	128
SRP_Print_DefaultJobSetup	L	512
SRP_Print_4DJobSetup	L	1024
SRP_Print_AskPageSetup	L	4096
SRP_Print_AskJobSetup	L	8192
SRP_Print_NoProgress	L	16384
SRP_Print_NoDefaultPrinter	L	32768
SRP_Print_WinPDFNoFonts	L	65536
SRP_Print_MacUseDestination	L	131072
SRP_Print_Default	L	12288
SRP_Print_WinOpenCreatedFile	L	262144
SRP_Print_WinPDFPreview	L	524288

See also Print Flags.

Legacy constants

SR Pro Event Codes

Constant	Type	Value
SR Zoom Area	L	11
SR UnZoom Area	L	12
SR Zoom Area to Back	L	13
SR Orig Area to Back	L	14
SR Area Closing	L	15
SR Editor Mode	L	20
SR Preview Mode	L	21
SR Preview First Page	L	30
SR Preview Previous Page	L	31
SR Preview Next Page	L	32
SR Preview Last Page	L	33
SR Preview Print Page	L	34
SR Close Preview	L	35
SR Menu Item Selected Before	L	40
SR Menu Item Selected After	L	41

SR Pro Menu IDs

Constant	Type	Value
SR MenuItem New	L	101
SR MenuItem Open	L	102
SR MenuItem Close	L	103
SR MenuItem Save	L	104
SR MenuItem Save As	L	105
SR MenuItem Print To Disk	L	106
SR MenuItem Preview	L	107
SR MenuItem Page Setup	L	108
SR MenuItem Print	L	109
SR MenuItem Zoom	L	110
SR MenuItem Undo	L	201
SR MenuItem Cut	L	202
SR MenuItem Copy	L	203
SR MenuItem Paste	L	204
SR MenuItem Clear	L	205
SR MenuItem Select All	L	206
SR MenuItem Duplicate	L	207
SR MenuItem Modify Object	L	208
SR MenuItem Change Object	L	209
SR MenuItem Position Object	L	210

Constant	Type	Value
SR MenuItem Modify Section	L	211
SR MenuItem Position Sections	L	212
SR MenuItem Activate Section	L	213
SR MenuItem Bring To Front	L	214
SR MenuItem Bring Forward	L	215
SR MenuItem Send To Back	L	216
SR MenuItem Send Backwards	L	217
SR MenuItem Use Physical Page	L	301
SR MenuItem Use Printable Area	L	302
SR MenuItem Rulers	L	303
SR MenuItem Ruler Units	L	304
SR MenuItem Grid	L	305
SR MenuItem Guides	L	306
SR MenuItem Lock Guides	L	307
SR MenuItem Sections	L	308
SR MenuItem Lock Sections	L	309
SR MenuItem Margins	L	310
SR MenuItem Object Borders	L	311
SR MenuItem Show Object Border	L	312
SR MenuItem Show Object Alias	L	313
SR MenuItem Main Table	L	401
SR MenuItem Select Records	L	402
SR MenuItem Order Records	L	403
SR MenuItem Print Sections	L	404
SR MenuItem Scripts	L	405
SR MenuItem Header	L	1001
SR MenuItem SubHeader1	L	1002
SR MenuItem SubHeader2	L	1003
SR MenuItem SubHeader3	L	1004
SR MenuItem SubHeader4	L	1005
SR MenuItem SubHeader5	L	1006
SR MenuItem SubHeader6	L	1007
SR MenuItem Body	L	1008
SR MenuItem SubTotal6	L	1009
SR MenuItem SubTotal5	L	1010
SR MenuItem SubTotal4	L	1011
SR MenuItem SubTotal3	L	1012
SR MenuItem SubTotal2	L	1013
SR MenuItem SubTotal1	L	1014
SR MenuItem Total	L	1015
SR MenuItem Footer	L	1016
SR MenuItem Ruler Points	L	1101
SR MenuItem Ruler Millimeter	L	1102

Constant	Type	Value
SR MenuItem Ruler Inches	L	1103
SR MenuItem Line Hair	L	5001
SR MenuItem Line 1	L	5002
SR MenuItem Line 2	L	5003
SR MenuItem Line 3	L	5004
SR MenuItem Line 4	L	5005
SR MenuItem Line 5	L	5006
SR MenuItem Line 6	L	5007
SR MenuItem Line 7	L	5008
SR MenuItem Line 8	L	5009
SR MenuItem Font Size 6	L	6001
SR MenuItem Font Size 7	L	6002
SR MenuItem Font Size 8	L	6003
SR MenuItem Font Size 9	L	6004
SR MenuItem Font Size 10	L	6005
SR MenuItem Font Size 11	L	6006
SR MenuItem Font Size 12	L	6007
SR MenuItem Font Size 14	L	6008
SR MenuItem Font Size 18	L	6009
SR MenuItem Font Size 24	L	6010
SR MenuItem Font Size 36	L	6011
SR MenuItem Font Size Smaller	L	6012
SR MenuItem Font Size Larger	L	6013
SR MenuItem Font Size Other	L	6014
SR MenuItem Start Script	L	7001
SR MenuItem Body Script	L	7002
SR MenuItem End Script	L	7003

SR Pro Sections

Constant	Type	Value
SR Num Sections	L	16
SR All Sections	L	65535
SR Print Defined Sections	L	-1
SR Section Header	L	0
SR Section SubHeader1	L	1
SR Section SubHeader2	L	2
SR Section SubHeader3	L	3
SR Section SubHeader4	L	4
SR Section SubHeader5	L	5
SR Section SubHeader6	L	6
SR Section Body	L	7
SR Section SubTotal6	L	8
SR Section SubTotal5	L	9
SR Section SubTotal4	L	10
SR Section SubTotal3	L	11
SR Section SubTotal2	L	12
SR Section SubTotal1	L	13
SR Section Total	L	14
SR Section Footer	L	15
SR Section Header Mask	L	1
SR Section SubHeader1 Mask	L	2
SR Section SubHeader2 Mask	L	4
SR Section SubHeader3 Mask	L	8
SR Section SubHeader4 Mask	L	16
SR Section SubHeader5 Mask	L	32
SR Section SubHeader6 Mask	L	64
SR Section Body Mask	L	128
SR Section SubTotal6 Mask	L	256
SR Section SubTotal5 Mask	L	512
SR Section SubTotal4 Mask	L	1024
SR Section SubTotal3 Mask	L	2048
SR Section SubTotal2 Mask	L	4096
SR Section SubTotal1 Mask	L	8192
SR Section Total Mask	L	16384
SR Section Footer Mask	L	32768

SR Pro Errors

Constant	Type	Value
SR Invalid Report Data	L	-8000
SR Cannot Create BLOB	L	-8001
SR Invalid Array Type	L	-8002
SR User Cancelled	L	-8003
SR No Selected Objects	L	-8004
SR Unknown Std Variable	L	-8005
SR Bad Menu ID	L	-8006
SR Bad Parameter	L	-8007
SR Invalid Area	L	-8008
SR Damaged Report	L	-8888
SR Editor Item ID Invalid	L	-2000
SR Editor Section Not Active	L	-2001
SR Editor Sect Position Invalid	L	-2002
SR Editor Cannot Disable Sects	L	-2003
SR Editor Invalid Sect Option	L	-2004
SR Editor Incompatible Obj Type	L	-2005

SR Pro Editor Codes

Constant	Type	Value
SR Editor Create Object	L	1
SR Editor Modify Object	L	2
SR Editor Modify Section	L	3
SR Editor Modify Object Script	L	4
SR Editor Modify Report Script	L	5
SR Editor Control Click Object	L	6
SR Editor Click Object	L	7
SR Editor Selection Changed	L	8
SR Editor Object Deleted	L	9
SR Object Type Text	L	1
SR Object Type Field	L	2
SR Object Type Variable	L	3
SR Object Type Line	L	4
SR Object Type Rectangle	L	5
SR Object Type Circle	L	6
SR Object Type Picture	L	8
SR Start Report Script	L	1
SR Body Report Script	L	2
SR End Report Script	L	3
SR Use Section Always	L	1
SR Use Section On Break	L	2
SR Use Section On First Page	L	3

Constant	Type	Value
SR Use Section On Second Page	L	4
SR Section Break On Field	L	1
SR Section Break On Variable	L	2
SR Section Break On Array	L	3
SR Section Throw Page None	L	1
SR Section Throw Page Before	L	2
SR Section Throw Page After	L	3
SR Section Throw Page Min Space	L	4
SR Section Keep On One Page	L	1
SR Section Adjust At Print Time	L	2
SR Obj Flag Left Line	L	1
SR Obj Flag Top Line	L	2
SR Obj Flag Right Line	L	4
SR Obj Flag Bottom Line	L	8
SR Obj Flag All Lines	L	15
SR Obj Flag Fixed Horizontal	L	16
SR Obj Flag Fixed Vertical	L	32
SR Obj Flag Grow Horizontal	L	64
SR Obj Flag Grow Vertical	L	128
SR Obj Flag Variable Width	L	256
SR Obj Flag Variable Height	L	512
SR Obj Flag Replace If Empty	L	1024
SR Obj Flag Record Calc Value	L	2048
SR Obj Flag Show Calc Value	L	4096
SR Obj Flag Repeating Object	L	8192
SR Obj Flag Repeat Vertically	L	16384
SR Obj Flag Repeat Horizontally	L	32768
SR Obj Flag Replace Row If Empt	L	65536
SR Variable Type Variable	L	1
SR Variable Type Array Auto	L	2
SR Variable Type Array Element	L	3
SR Calculation Type None	L	0
SR Calculation Type Total	L	1
SR Calculation Type Min	L	2
SR Calculation Type Average	L	3
SR Calculation Type Max	L	4
SR Pict Format Normal	L	0
SR Pict Format Centered	L	1
SR Pict Format Scaled To Fit	L	2
SR Pict Format Scaled Prop	L	3
SR Pict Format Scaled Prop Cent	L	4
SR Iterations Main Table	L	1
SR Iterations Fixed	L	2

Constant	Type	Value
SR Iterations Variable	L	3
SR Iterations Array	L	4
SR Property All	L	-1
SR Property Name	L	1
SR Property Position	L	2
SR Property Type	L	4
SR Property Options	L	8
SR Property Selected	L	16
SR Property Field	L	32
SR Property Variable Type	L	64
SR Property Calculation	L	128
SR Property Rows Cols	L	256
SR Property Repeat Offsets	L	512
SR Attribute All	L	-7169
SR Attribute Font Name	L	1
SR Attribute Fore Color	L	2
SR Attribute Back Color	L	4
SR Attribute Font Size	L	8
SR Attribute Font Style	L	16
SR Attribute Fore Pattern	L	32
SR Attribute Back Pattern	L	64
SR Attribute Justification	L	128
SR Attribute Line Thickness	L	256
SR Attribute Format	L	512
SR Attribute 4D Fore Color	L	1024
SR Attribute 4D Back Color	L	2048
SR Attribute No Adjust	L	4096
SR PowerMenu Tables	L	0
SR PowerMenu Fields	L	1
SR PowerMenu Variables	L	2
SR PowerMenu Commands	L	3
SR PowerMenu Break Object Type	L	5
SR PowerMenu Variable Type	L	6
SR PowerMenu Format	L	7
SR PowerMenu Color	L	11
SR PowerMenu Pattern	L	12
SR PowerMenu Line	L	13
SR Position At Front	L	-1
SR Position At End	L	-2
SR Position Forward	L	-3
SR Position Backward	L	-4
SR All Objects	L	0
SR Selected Objects	L	1

Constant	Type	Value
SR Tool Arrow	L	0
SR Tool Text	L	1
SR Tool Field	L	2
SR Tool Variable	L	3
SR Tool Line	L	4
SR Tool Rectangle	L	5
SR Tool Circle	L	6

SR Pro Options

Constant	Type	Value
SR Generic Option Query	L	-1
SR Generic Option Set Off	L	0
SR Generic Option Set On	L	1
SR Print Option Validate	L	1
SR Print Option Job Dialog	L	2
SR Print Option No Progress	L	4
SR Print Option 4DPrintSettings	L	8
SR Print Option 4DPageSetup	L	16
SR Print Option 4DJobSetup	L	32
SR Print Option UseDefPrinter	L	64
SR Save Option File Dialog	L	1
SR DoCommand Use Custom Proc	L	0
SR DoCommand Ignore Custom Proc	L	1
SR MenuItem Count Items	L	-1
SR MenuItem Query	L	0
SR MenuItem Set Text	L	1
SR MenuItem Set Status	L	2
SR MenuItem Set Mark	L	4
SR MenuItem Set 4D Method	L	8
SR PrintToDisk File Dialog	L	1
SR PrintToDisk Output HTML	L	1
SR PrintToDisk Body Only	L	2
SR PrintToDisk Static Text	L	4
SR PrintToDisk TopLeft Order	L	8
SR PrintToBLOB Output HTML	L	1
SR PrintToBLOB Body Only	L	2
SR PrintToBLOB Static Text	L	4
SR PrintToBLOB TopLeft Order	L	8
SR PrintToPict Validate	L	1
SR PrintToPict Job Dialog	L	2
SR PrintToPict No Progress	L	4
SR PrintToPict 4DPrintSettings	L	8

Constant	Type	Value
SR PrintToPict 4DPageSetup	L	16
SR PrintToPict 4DJobSetup	L	32
SR MainTable Show Dialog	L	-1
SR MainTable Query	L	0
SR MainTable Choose Table	L	1
SR MainTable Fixed Iteration	L	2
SR MainTable Variable	L	4
SR MainTable Array	L	8
SR Options Hide Zoom	L	1
SR Options Basic Interface	L	2
SR Options Hide HTML	L	4
SR Options Use Std Guides	L	8
SR Options Use Custom Struct	L	16
SR Options Hide Menubar	L	32
SR Options Hide Toolbar	L	64
SR Options Hide Hor Scrollbar	L	128
SR Options Hide Vert Scrollbar	L	256
SR FileType Mac	L	1
SR FileType Win	L	2
SR FileType Get Creator	L	-1
SR FileType Get Text Type	L	-2
SR FileType Get Doc Type	L	-4
SR FileType Set Creator	L	1
SR FileType Set Text Type	L	2
SR FileType Set Doc Type	L	4
SR AreaFormat Physical Page	L	1
SR AreaFormat Printable Area	L	2
SR SetAreaFormat Adjust Objects	L	1
SR SetAreaFormat Adjust Sects	L	2
SR SetAreaFormat Adjust Guides	L	4
SR Structure Physical	L	0
SR Structure Virtual	L	1
SR Structure Area Structure	L	2
SR Structure Get Invisible	L	4
SR Structure Get SubTables	L	8
SR Structure Get Indexed Only	L	16
SR Structure Get Empty Tables	L	32
SR Structure Sort By Name	L	64
SR Structure No MenuID	L	128
SR Structure Mark Indexed	L	256
SR Structure Mark Invisible	L	512
SR Commands By First Char	L	0
SR Commands By Theme	L	1

Constant	Type	Value
SR Commands Area Commands	L	2
SR Commands Sort Second Level	L	4
SR Commands Sort First Level	L	8
SR Preview Wait For Close	L	1
SR Preview Zoom In	L	2
SR PrintDestination Printer	L	1
SR PrintDestination File	L	2
SR PrintDestination Fax	L	3
SR PrintDestination Preview	L	4
SR PrintDestination PDF	L	5

Copyrights and Trademarks

All trade names referenced in this document are the trademark or registered trademark of their respective holder.

4D and 4D Server are trademarks of 4D SAS.

Windows is a trademark of Microsoft Corporation.

Macintosh, MacOS and MacOS X are trademarks of Apple, Inc.

Index

Symboles

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X