

SYSPRO Reporting Services API

Developer Guide

Published: March 2018





Contents

SYSPRO Reporting Services API.....	3
Getting Started.....	4
Prerequisites.....	4
Restrictions.....	4
Known limitations.....	4
Deploy and Use.....	5
Installing / starting the SRS Host Service.....	5
Configuring server-side reporting in SYSPRO 7.....	6
Configuring server-side reporting in SYSPRO 8.....	6
How to use the API.....	6
How to add a reference to the client library assembly.....	7
API Methods.....	7
Sample XML.....	8
Sample Code.....	10
 Additional resources.....	 15

SYSPRO Reporting Services API

SYSPRO's Reporting API allows developers to produce SRS documents using the SYSPRO Reporting Service Server infrastructure.

At a very high-level, the SRS Service is the engine that produces SRS documents. This feature has now been exposed to developers via an API. The API allows developers to securely query the SYSPRO database and produce the required document. Once the document is produced, it can be:

- stored for later use,
- printed, and
- emailed.

Developers can also choose to access the document via the report queue or via the API for further automation.

Once the documents are generated, they are added to the document printing queue from where they can be viewed, executed and managed (**SYSPRO Programs > SRS Documents > Document Setup > SRS Document Queue**).



The queue item has a different destination code to indicate that it came from the API and not from within SYSPRO.

Getting Started

Prerequisites

Before using the SRS API, ensure that SRS server-side printing is working correctly in SYSPRO.

The following is required:

Server

- For SYSPRO 7 - Reporting Host Service (CR 2013) Port 31
- For SYSPRO 8 - Reporting Host Service
- Crystal Reports Server Embedded (CR 2013) SP8
- SAP BusinessObjects BI platform .Net SDK Redistributable 64-bit 4.2. SP3 (only for SYSPRO 8)

Client

- Browser pop-ups must be enabled



This is only applicable for previewing PDFs in a web browser application.

SYSPRO Configuration Requirements

- Server side reporting (configured within the SYSPRO **Program List > Administration > General Setup > System Setup** program)
- e.net service details (configured within the SYSPRO **Program List > Administration > General Setup > System Setup** program)

Restrictions

- When emailing a document, only one email address can be entered in the **To** and **CC** lines.

Known limitations

The following are known limitations, which will be addressed during the lifetime of SYSPRO 8:

- Batch printing is currently not supported.
- Documents that are printed using this architecture are currently not added to the document archive.

The generated documents are stored in the SRS document queue used for server side printing and will be visible from the **SRS Document Queue** program. However, they are not added to the document archive.

There will be no PDF file in the **SRS Document Printing** archive folder, nor will the document be visible in the **Document Archive Viewer**.

- This functionality is limited to the following Sales Order document types:
 - Invoice
 - Delivery note
 - Order acknowledgement



Support for other documents and reports will be made available in the future.

Deploy and Use

Installing / starting the SRS Host Service

The following steps describe how to install the SRS Host Service.

SYSPRO 8

In SYSPRO 8, the Host Service install is called `SYSPRO.8.REPORTING.HOST.SERVICE.EXE`.

The Host Service is deployed with the **SYSPRO Installer** and can be installed automatically when SYSPRO 8 is installed.

We recommend stopping the Host Service before any port updates are done and starting it again afterwards. This will release any e.net business objects which may be held by the service.



You don't need to reinstall the service, except if an updated and/or improved version of the service is released.

SYSPRO 7

In SYSPRO 7, the Host Service is called `SYSPROReportingHostService2013.exe` and has to be installed.

1. To install the `SYSPROReportingHostService2013.EXE`, please refer to the [Installing and configuring SRS server-side printing](#) document that is located in the **InfoZone**.

2. . Run the `SYSPROReportingHostService2013.exe` executable.

You can obtain the file from the **Ports and Downloads** section of the **InfoZone**. Otherwise, contact your local reseller for more information.

3. During installation you will be required to configure settings for the service.

Service Configuration screen:

Method	Description
SYSPRO Application Server Instance	This indicates the default instance of SYSPRO with which the service will communicate. SYSPRO instances are reflected in your Windows Registry to identify the Base folder of your SYSPRO install, where the necessary .dll and .exe files are located.
SYSPRO RAS Management Service Port	This is the port number used for SYSPRO's RAS Management Service. This is used to balance the load of server-side printing requests received.
SYSPRO e.net WCF Service Endpoint	This is the address to the SYSPRO e.net WCF Service and is only required if you are installing on a server other than the SYSPRO Application Server (defaults to <code>net.tcp://localhost:90/SYSPROWCFService</code>).

Printing Service Credentials screen (for printer access)::

Method	Description
Domain	The name of the network to which you are connecting.
User name	The user login credentials used to access the network.
Password	The password credentials used to access the network.
Enter the users profile temp folder	This is the location of the temporary folder defined against the user's profile.



We recommend stopping the service before any port updates are done and starting it again afterwards. This will release any e.net business objects which may be held by the service.



You don't need to reinstall the service, except if an updated and/or improved version of the service is released.


Configuring server-side reporting in SYSPRO 7

The following steps describe how to configure server-side reporting.

1. Open the **System Setup** program and select the **Reporting** tab (**Program List > Administration > General Setup > System Setup**).
2. On this screen, select the following:
 - At the **Reporting configuration** select **Server-side reporting using SQL**.
 - At the **Crystal Version** select **Crystal 2013**.
 - At the **SQL Server name** field enter the name of the SQL Server instance that contains the _SRS database.
 - At the **Reporting authentication** select the type of authentication you will use.
 -  If you select **Windows authentication**, ensure that the user name supplied has permission to create a database, and to read/write data to it.
 - At the **Reporting service** field, enter the name of the reporting service followed by the service port. The default service port is 1979.
 -  Don't use the localhost name, as this will not work from a client workstation.
3. The first time that the Host Service is started, the Admin operator is copied to the **__SRS operator code**.
4. Select **Save and Exit** to save your changes.

Configuring server-side reporting in SYSPRO 8

The following steps describe how to configure server-side reporting.

1. Open the **System Setup** program and select the **Reporting** tab (**Program List > Administration > General Setup > System Setup**).
2. On this screen, select the following:
 - At the **Reporting configuration** select **Server-side reporting using SQL**.
 - At the **SQL Server name** field enter the name of the SQL Server instance that contains the _SRS database.
 - At the **Reporting authentication** select the type of authentication you will use.
 -  If you select **Windows authentication**, ensure that the user name supplied has permission to create a database, and to read/write data to it.
3. The first time that the Host Service is started, the Admin operator is copied to the **__SRS operator code**.
4. Select **Save and Exit** to save your changes.

How to use the API

All API calls are done via the SRS library.

1. Create a new project and create the DLLs as a reference.

Use the base address from SYSPRO, when creating an instance of the class. (**SYSPRO > Reporting tab > Server-side config > Reporting Service**)

2. Use the five available methods to query and generate documents.

How to add a reference to the client library assembly

1. Open the solution.
2. Select the Solution Explorer (**View >Solution Explorer**).
3. Right-click on **References** in the Solution Explorer pane of the project and select **Add Reference**.
4. Select **Browse** and navigate to the **SYSPRO Base** directory.
5. Select **SYSPROSRSCientLibrary.dll** and select **Add**.
6. Make sure the assembly is checked in the list of assemblies, and select **OK**.

API Methods


The exposed API methods can be used in a number of ways:


- Authentication
- Controlling login and logout.
- Controlling which documents can be generated.
- Controlling the formats of the documents.

This allows customers with login credentials to access their data to generate documentation from third-party applications. External or third-party applications like Espresso can request the information from the API and, once received, apply it to generate the required documents.

The `SYSPROSRSDocumentAPI.dll` assembly (located in the `\Base` folder) allows an application like Espresso to call the Host Service in order to create the sales order documents.

The following API methods are exposed:

Method	Description
<code>AuthSusproUser</code>	This controls and authorizes the logging on of users. This method returns a session ID.
<code>AuthenticateSysproGUID</code>	This controls and authorizes access of users that are already logged in. This method returns a session ID.
<code>LogOffUserSession</code>	This logs off sessions of stand-alone applications.
<code>DetermineDocument Options</code>	<p>This provides a list of formats for the selected document type.</p> <p>This method returns XML with details of available documents and formats.</p> <p> It only returns SRS formats and looks at the control file (i.e. sales order control file), to determine which formats are defined for SRS according to the governing business rules.</p> <p>For the Sales Order Document Printing functionality, this method returns the following that can be used to determine which document types can be printed:</p> <ul style="list-style-type: none">• Sales order number• Sales order status

Method	Description
	<ul style="list-style-type: none"> Sales order flags
ProduceDocument	<p>This generates the document to print on the host server. When previewing the document, it is returned in a HEX encoded format of a PDF document.</p> <p> A third-party developer will have to convert the content back to ASCII format, before saving it to disk.</p> <p>The input XML is passed to the business object, which in turn communicates with the Document Print program. The Host Service uses the XML returned from the Document Print program, to generate the document using the selected format.</p> <p>The following codes are used for the documents types:</p> <ul style="list-style-type: none"> I - invoice D - delivery note O - order acknowledgment



The functionality of these methods may be extended in future releases.

Sample XML

The following XML samples are provided to assist the developer.

Get document details

Parameter	Sample XML
xmlParam	None
xmlIn	<pre> <Query> <Option> <Function>GETDOCDET</Function> </Option> <Filter> <OrderNumber FilterType='S' FilterValue='791' / > </Filter> </Query> </pre>
xmlout	<pre> <DocumentControl Language="05" Language2="EN" CssStyle="" DecFormat="1" DateFormat="01" Role="01" Version="8.0.001" OperatorPrimaryRole="" > <DocumentInformation> <SalesOrder>000791</SalesOrder> <OrderStatus>9</OrderStatus> <Translated_OrderStatus>Complete</ Translated_OrderStatus> <ActiveFlag>N</ActiveFlag> <Translated_ActiveFlag>No</Translated_ActiveFlag> <CancelledFlag /> <Translated_CancelledFlag>No</ Translated_CancelledFlag> <SODocumentType>O</SODocumentType> </pre>

Parameter	Sample XML
	<pre> <Translated_SODocumentType>Order</ Translated_SODocumentType> <CanPrintAcknowledgement>>false</ CanPrintAcknowledgement> <CanRePrintAcknowledgement>>false</ CanRePrintAcknowledgement> <CanPrintDeliveryNote>>false</CanPrintDeliveryNote> <CanRePrintDeliveryNote>>false</ CanRePrintDeliveryNote> <CanGenerateInvoice>>false</CanGenerateInvoice> <Documents> <Invoice> <InvoiceNumber>100506</InvoiceNumber> <InvoiceSource>0</InvoiceSource> <Translated_InvoiceSource>Order</ Translated_InvoiceSource> <DateLastInvPrt>2015-04-08</DateLastInvPrt> </Invoice> </Documents> <Formats> <Format> <DocumentType>Order Acknowledgement</ DocumentType> <FormatCode>0</FormatCode> <FormatName>Order Acknowledgemen</FormatName> </Format> <Format> <DocumentType>Delivery Note</DocumentType> <FormatCode>0</FormatCode> <FormatName>Delivery Note</FormatName> </Format> <Format> <DocumentType>Invoice</DocumentType> <FormatCode>0</FormatCode> <FormatName>Invoice</FormatName> </Format> </Formats> </DocumentInformation> </DocumentControl> </pre>

Produce document

Parameter	Sample XML
xmlParam	<pre> <DocumentControl> <DocumentType>SalesOrder</DocumentType> <Print>False</Print> <Email>False</Email> <Preview>True</Preview> <XmlOnly>False</XmlOnly> <PrinterDetails> <PrinterName /> <PrintCopies>1</PrintCopies> <PrintCollate>True</PrintCollate> </PrinterDetails> <EmailDetails> <EmailFromAddress /> <EmailToAddress /> <EmailCcAddress /> <EmailSubject /> <EmailBodyText /> </EmailDetails> </pre>

Parameter	Sample XML
	<code></DocumentControl></code>
xmlIn	<pre> <Query> <Option> <Function>ONLINE</Function> <DocumentType>I</DocumentType> <Format>0</Format> <Reprint>Y</Reprint> </Option> <Filter> <OrderNumber FilterType='S' FilterValue='000791' /> <InvoiceNumber FilterType='S' FilterValue='100506' /> </Filter> </Query> </pre>
xmlout	<pre> <Query> <Document> <Status /> <DocumentGuid/> <ErrorMessage/> <StatusPreview/> <DocumentHex>HEX Encoded PDF file</ DocumentHex > </Document> </Query> </pre>

Sample Code

The following sample code is provided to assist the developer in using the API.



All sample code is provided in C#.

Create the SYSPROSRClient object

Create the SYSPROSRClient object

```

// Replace "localhost:20140" with the endpoint of your
// e.net load balancer service
// See the E.Net Service Details tab in System Setup
SYSPROSRClientLibrary.SYSPROSRClient _sysproSRClient =
    new SYSPROSRClientLibrary.SYSPROSRClient("localhost:20140");

```

```

// Replace "localhost:20140" with the endpoint of your e.net loadbalancer
// service
SYSPROSRClientLibrary.SYSPROSRClient _sysproSRClient = new
SYSPROSRClientLibrary.SYSPROSRClient("localhost:20140");

```

Authenticate to the service using operator and company code

Authenticate to the service using operator and company code

```
// The following assumes that you have set up the method arguments
// with appropriate values
string sessionId = _sysproSRSCient.AuthenticateSYSPROUser(
    OperatorCode, OperatorPassword, CompanyCode, CompanyPassword);
```

```
string sessionId = _sysproSRSCient.AuthenticateSYSPROUser({OperatorCode},
    {OperatorPassword}, {CompanyCode}, {CompanyPassword});
```

Authenticate to the service you use

Authenticate to the service you use

```
// The following assumes that you have set up the method argument
// with an appropriate value from a previous connection to SYSPRO
string sessionId = _sysproSRSCient.AuthenticateSYSPROGuid(GUID);
```

```
string sessionId = _sysproSRSCient.AuthenticateSYSPROGuid({GUID});
```

Determine the available document options

Determine the available document options

```
StringBuilder xmlIn = new StringBuilder();
xmlIn.Append("<Query>");
xmlIn.Append("<Option>");
xmlIn.Append("<Function>GETDOCDET</Function>");
xmlIn.Append("<IncludeFormatDetails>Y</IncludeFormatDetails>");
xmlIn.Append("</Option>");
xmlIn.Append("<Filter>");
xmlIn.Append("<OrderNumber FilterType='S' ");
// Where 791 is the Sales Order number
xmlIn.AppendFormat("FilterValue='{0}' />", 791);
xmlIn.Append("</Filter>");
xmlIn.Append("</Query>");
string documentOptions = _sysproSRSCient.DetermineDocumentOptions(
    sessionId, "SalesOrder", xmlIn.ToString());
// See documentation for sample XML out
```

```
StringBuilder xmlIn = new StringBuilder();
xmlIn.Append("<Query>");
xmlIn.Append("<Option>");
xmlIn.Append("<Function>GETDOCDET</Function>");
xmlIn.Append("<IncludeFormatDetails>Y</IncludeFormatDetails>");
xmlIn.Append("</Option>");
xmlIn.Append("<Filter>");
xmlIn.AppendFormat("<OrderNumber FilterType='S' FilterValue='{0}' />",
791); // Where 791 is the Sales Order number
xmlIn.Append("</Filter>");
xmlIn.Append("</Query>");

string documentOptions =
_sysproSRSCient.DetermineDocumentOptions(_sessionId, "SalesOrder",
xmlIn.ToString());
// See documentation for sample XML out
```

Produce the document:

Produce the document

```
StringBuilder xmlParam = new StringBuilder();
xmlParam.Append("<DocumentControl>");
xmlParam.Append("<DocumentType>SalesOrder</DocumentType>");
xmlParam.Append("<Print>False</Print>");
xmlParam.Append("<Print>False</Print>");
xmlParam.Append("<Preview>True</Preview>");
xmlParam.Append("<XmlOnly>False</XmlOnly>");
xmlParam.Append("<PrinterDetails>");
xmlParam.Append("<PrinterName />");
xmlParam.Append("<PrintCopies>1</PrintCopies>");
xmlParam.Append("<PrintCollate>True</PrintCollate>");
xmlParam.Append("</PrinterDetails>");
xmlParam.Append("<EmailDetails>");
xmlParam.Append("<EmailFromAddress />");
xmlParam.Append("<EmailToAddress />");
xmlParam.Append("<EmailCcAddress />");
xmlParam.Append("<EmailSubject />");
xmlParam.Append("<EmailBodyText />");
xmlParam.Append("</EmailDetails>");
xmlParam.Append("</DocumentControl>");

StringBuilder xmlIn = new StringBuilder();
xmlIn.Append("<Query>");
xmlIn.Append("<Option>");
// This is always 'ONLINE'
xmlIn.Append("<Function>ONLINE</Function>");
// See Document Type Codes
xmlIn.Append("<DocumentType>I</DocumentType>");
// Format code selected to print with
// See the output from DetermineDocumentOptions
xmlIn.Append("<Format>0</Format>");
// Print/Reprint flag
// See the output from DetermineDocumentOptions
xmlIn.Append("<Reprint>Y</Reprint>");
xmlIn.Append("</Option>");
xmlIn.Append("<Filter>");
xmlIn.Append("<OrderNumber FilterType='S' ");
// Where 791 is the Sales Order number
xmlIn.AppendFormat("FilterValue='{0}'>", 791);
xmlIn.Append("<InvoiceNumber FilterType='S' ");
// Where 100506 is the Invoice number
xmlIn.AppendFormat("FilterValue='{0}'>", 100506);
xmlIn.Append("</Filter>");
xmlIn.Append("</Query>");
string xmlOut = _sysproSRSCClient.ProduceDocument(
    sessionId, xmlParam.ToString(), xmlIn.ToString());
// See documentation for sample XML out
```

```
StringBuilder xmlParam = new StringBuilder();
xmlParam.Append("<DocumentControl>");
xmlParam.Append("<DocumentType>SalesOrder</DocumentType>");
xmlParam.Append("<Print>False</Print>");
xmlParam.Append("<Print>False</Print>");
xmlParam.Append("<Preview>True</Preview>");
xmlParam.Append("<XmlOnly>False</XmlOnly>");
xmlParam.Append("<PrinterDetails>");
```

```

xmlParam.Append("<PrinterName />");
xmlParam.Append("<PrintCopies>1</PrintCopies>");
xmlParam.Append("<PrintCollate>True</PrintCollate>");
xmlParam.Append("</PrinterDetails>");
xmlParam.Append("<EmailDetails>");
xmlParam.Append("<EmailFromAddress />");
xmlParam.Append("<EmailToAddress />");
xmlParam.Append("<EmailCcAddress />");
xmlParam.Append("<EmailSubject />");
xmlParam.Append("<EmailBodyText />");
xmlParam.Append("</EmailDetails>");
xmlParam.Append("</DocumentControl>");

StringBuilder xmlIn = new StringBuilder();
xmlIn.Append("<Query>");
xmlIn.Append("<Option>");
// This is always 'ONLINE'
xmlIn.Append("<Function>ONLINE</Function>");
// See Document Type Codes
xmlIn.Append("<DocumentType>I</DocumentType>");

// Format code selected to print with - see output from
DetermineDocumentOptions
xmlIn.Append("<Format>0</Format>");

xmlIn.Append("<Reprint>Y</Reprint>");
// Print/Reprint flag - see output from DetermineDocumentOptions
xmlIn.Append("</Option>");
xmlIn.Append("<Filter>");
xmlIn.AppendFormat("<OrderNumber FilterType='S' FilterValue='{0}' />", 791); // Where 791 is the Sales Order number
xmlIn.AppendFormat("<InvoiceNumber FilterType='S' FilterValue='{0}' />", 100506); // Where 100506 is the Invoice number
xmlIn.Append("</Filter>");
xmlIn.Append("</Query>");

string xmlOut = _sysproSRSCient.ProduceDocument(sessionId,
xmlParam.ToString(), xmlIn.ToString());
// See documentation for sample XML ou

```

Retrieve the document content from the HEX encoded string from the output XML

Retrieve the document content from the HEX encoded string from the output XML

```
string docHex = "";
XDocument xOut = XDocument.Parse(xmlOut);
if (xOut.Element("Query").Element("Document") != null)
{
    XElement xDocElemnt = xOut.Element("Query").Element("Document");
    if (xDocElemnt.Element("DocumentHex") != null)
    {
        docHex = xDocElemnt.Element("DocumentHex").Value;
    }
}
byte[] docByte = Hex_to_ByteArray(docHex)

/// <summary>
/// Function that converts a HEX encoded string to an array
/// of unsigned integers that represent the bytes
/// </summary>
public static byte[] Hex_to_ByteArray(string HexString)
{
    // Get the number of total characters in the string
    int NumberChars = HexString.Length;

    // Each byte is derived from 2 characters in the input
    // that represent the HEX value of the byte
    byte[] bytes = new byte[NumberChars / 2];

    for (int i = 0; i < NumberChars; i += 2)
    {
        // The 16 in the following conversion indicates that this
        // is converting from Hex, or base 16.
        bytes[i / 2] = Convert.ToByte(HexString.Substring(i, 2), 16);
    }
    return bytes;
}
```

```
// Document Content is HEX encoded as it may contain invalid XML
characters, i.e. NULL characters
string hexString = {Get content from XML};
hexString = hexString.Replace(" ", "");
byte[] arrASCII = new byte[hexString.Length / 2];
for (int i = 0; i < hexString.Length; i += 2)
{
    arrASCII[i / 2] = (byte)Convert.ToByte(hexString.Substring(i, 2), 16);
}
```

Logoff from the SYSPRO session

Logoff from the SYSPRO session

```
// If you authenticated using a OperatorCode and
// CompanyCode then you might want to logoff.
_sysproSRSCient.LogoffUserSession(sessionId);
```

```
// If you authenticated using a OperatorCode and CompanyCode then you might
want to logoff.
_sysproSRSCient.LogoffUserSession(sessionId);
```

Additional resources

Reference Guides

SYSPRO's Reference Guides are primarily module-based and cover aspects of the user interface at program level. This includes detailed field and function explanations as well as notes and warnings regarding the usage of an application program within SYSPRO. These guides also extend to feature topics within the system (e.g. Tax, Security, Language Translation, etc.) and include implementation considerations. Please refer to the SYSPRO [InfoZone](http://infozone.syspro.com/support) for details on how to obtain these guides (<http://infozone.syspro.com/support>).

Support

SYSPRO's [InfoZone](#) provides up-to-date information about the product as well as more advanced tutorials for registered users.

Newsletter

As part of SYSPRO's ongoing commitment to keeping you informed about the latest product developments, a regular newsletter is distributed to the SYSPRO community. The newsletter covers many aspects of SYSPRO ranging from product enhancements to support-related information, known issues and useful tips. You can subscribe to this newsletter from the SYSPRO [InfoZone](#).

Forums

SYSPRO provides a number of active on-line [forums](#) for you to engage in various discussions about the product.

Contact us



The Technical Authoring team comprises: Freya Nell, Carol Hart, Monique MacNaught and Even Nasset. Send us your comments to help us improve the standard of our reference guides.



Emails can be sent to Development-Authors@za.syspro.com.