

Reporting Services

Developer API Guide

Version 1.0 · For external developer use



© 2026 Syspro (Pty) Ltd. All rights reserved.

Intended for external developers integrating with the Syspro Reporting Services API. Information is subject to change without notice.

1. Overview

Syspro Reporting Services (SRS) is the reporting API for the Syspro ERP platform. It processes two kinds of requests through the same job pipeline: standard reports, identified by setting `templateType` to "Report", and stationery documents such as invoices or statements, identified by setting `templateType` to the document type registered in SRSPM2 (SRS Document Maintenance). For both request types, SRS orchestrates data retrieval via the Syspro e.net layer, renders output through the Wyn report engine, and delivers results through configurable post-processing extensions.

SRS does not render reports itself — it orchestrates. Wyn renders. Syspro e.net supplies data, schemas, and templates. SQL Server persists the job queue and output. The Application Gateway sits in front of it all and routes external traffic.

Architecture

Component	Role
Syspro e.net (SOAP/WCF)	User auth, report metadata, templates, data retrieval, archiving
Wyn Report Engine (REST)	Template upload, report rendering, PDF/Excel/Word export
SQL Server	Job queue, completed reports, schedules, audit trail
Application Gateway	Reverse proxy — all external traffic routes through it

Base URL

All API calls in this guide are relative to:

```
https://{gateway-host}/syspro/srs
```

The gateway host is configured during Syspro installation. Contact your system administrator if you do not know the value for your environment.

2. Authentication

SRS uses JWT Bearer authentication. Tokens are obtained by exchanging a valid Syspro e.net session token. Once issued, a token is reusable for its configured lifespan.

Obtain a token

POST your e.net session token to the logon endpoint.

```
POST /api/Authenticate/LogonWithSysproEnetToken

Request body:
{
  "applicationId": "Syspro",
  "applicationSecret": "Syspro",
  "sysproEnetToken": "{your-enet-session-token}"
}

Response (200 OK):
{
  "sessionToken": "eyJ..."
}
```

Include the token in all subsequent requests:

```
Authorization: Bearer {sessionToken}
```

The e.net session token is obtained through the normal Syspro logon process. SRS does not perform the initial Syspro logon — it validates an existing session. `applicationId` and `applicationSecret` identify the calling application to SRS rather than the individual user; for most environments both are simply "Syspro", but check with the Syspro administrator for the values registered for your environment.

Validate a token

Confirm a token is still active without making a real API call.

```
GET /api/Authenticate/IsAuthenticated

Response 200 OK    - token is valid
Response 401      - token is invalid or expired
```

Query string token support

In browser-based scenarios where setting an Authorization header is not possible (for example, opening the template designer in an iframe), pass the token as a query string parameter:

```
?access_token={sessionToken}
```

 Do not expose access tokens in URLs in production applications. Use this only for browser-embedded designer flows.

3. Report discovery

Before submitting a job, you need the report identifier and the parameters it accepts.

List available reports

Returns all reports accessible to the authenticated user in the current company context.

```
GET /api/Report/AvailableReports
```

Report IDs are the report name in lowercase with spaces replaced by underscores. "List Of Bank Balances" becomes "list_of_bank_balances". Use this ID in all subsequent calls. This endpoint only returns standard reports - it does not list stationery documents (e.g. invoices, statements). Document types are configured separately in SRSPM2 (SRS Document Maintenance); see Document job example below for how to submit a document request.

Get report parameters

Returns all parameters defined for a specific report. Use this to understand what input the report requires before submitting a job.

```
GET /api/Report/{reportName}/Parameters
```

For standard (BO-based) reports, parameters map to elements in the business object XML input — Options, Filters, and Keys. For third-party reports, parameters are read from the template itself.

Parameter structure:

```
[
  {
    "required": true,
    "label": "Period",
    "sysproSearchName": "string",
    "parameterId": "Query/Option/Period",
    "restrictedValues": {
      "1": "January",
      "2": "February"
    },
    "supportedFilters": ["Single", "All"],
    "type": "String",
    "defaultValue": "1",
    "dependentParameter": [],
    "selectAllValue": "0"
  }
]
```

Parameter fields

Field	Description
parameterId	Use as the id when building job input. For BO reports this is the XPath-style path, e.g. Query/Option/Period or Query/Filter/Bank.

Field	Description
label	Human-readable caption. Use this if building a UI from parameters.
type	Data type: String, Bool, Numeric, Date. Boolean values use true/false, not Y/N.
restrictedValues	If present, only these values are valid. Key = value to submit, Value = display label.
supportedFilters	Filter modes for this parameter: Single, Range, All, etc.
defaultValue	Pre-selected value where applicable.
required	Whether this parameter must be supplied.

 Parameters prefixed with Extensions/ are not report-specific inputs — they control post-processing behaviour (archive, export, email, print, notifications). See Section 6: Extensions.

Get document parameters

For stationary documents (invoices, factory documentation, etc.), use the document-specific endpoint:

```
GET /api/Report/{templateType}/DocumentParameter
```

4. Submitting jobs

Report and document requests are processed as jobs. A job is submitted with input parameters and optional extension instructions. SRS queues it, processes it asynchronously, and stores the result.

Process a report (synchronous)

Submit a job and wait for it to complete before the response is returned.

```
POST /api/Report/{reportName}/ProcessReport
```

Add a job to the queue (asynchronous)

The primary fire-and-forget endpoint. Submits a job, queues it for immediate processing, and returns a job ID. Use this for the vast majority of cases.

```
POST /api/ReportQueue
```

Deferred queuing (advanced)

Use this two-step approach only when you need to create a job record but hold it back from processing until a later point — for example, when building a batch or waiting for a precondition to be met.

```
// Step 1: Create the job in a held state
POST /api/Job/Pending

// Step 2: Release it for processing when ready
PUT /api/Job/jobs/mark-in-queue
```

Job request body

Both endpoints accept the same payload. The example below is for a standard report:

```
{
  "template": "list_of_bank_balances",
  "priority": "Low",
  "printCoverSheet": false,
  "reportTheme": "",
  "metaData": "",
  "languageCode": "EN",
  "templateType": "Report",
  "expiryDate": "2027-01-01T00:00:00.000Z",
  "datasourceId": "",
  "batchId": "",
  "jobStatusCode": "InQueue",
  "description": "Bank balances – January",
  "parameters": [
    {
      "id": "Query/Option/Month",
      "filterType": "Single",
      "value": ["1"]
    },
    {
      "id": "Query/Filter/Bank",
```

```

    "filterType": "All",
    "value": [""]
  }
]
}

```

Job request fields

Field	Type	Description
template	string	Report identifier (lowercase, underscores). Matches the id from /api/AvailableReports.
templateType	string	"Report" for standard reports. For documents: the document type, e.g. "Invoice" or "Factory Documentation".
priority	string	"Low", "Medium", or "High".
languageCode	string	ISO language code for report translation. Use "EN" for English.
printCoverSheet	bool	Whether to include a cover sheet in the generated report.
expiryDate	datetime	ISO 8601 timestamp. Completed report output is retained until this date.
parameters	array	List of input parameters. See structure below.
metaData	string	Passed to archive extensions when archiving documents.
datasourceId	string	Leave empty for standard jobs. Used when a pre-built datasource exists.
description	string	Free-text label. Used in notifications and filename placeholders.

Parameter input structure

Field	Description
id	The parameterId value from the parameters endpoint.
filterType	"Single" for a specific value. "Range" for from/to. "All" to include all (empty value list).
value	Array of strings. Single: one-element array. Range: [from, to]. All: [""].

Document job example

For stationary documents, set templateType to the Document Type configured in SRSPM2 (SRS Document Maintenance) - for example "Factory Documentation". Set template to the matching Document identifier registered against that Document Type in SRSPM2 (for example

"api_doc_template_id") - this is the key linked to a specific stationery format for that document type, not the .rdlx template file name. A Document Type can have several Document entries, each pointing to its own format and template file, so use the Document identifier for the exact format you want to run.

```
{
  "template": "api_doc_template_id",
  "priority": "Low",
  "templateType": "Factory Documentation",
  "languageCode": "EN",
  "printCoverSheet": false,
  "expiryDate": "2027-01-01T00:00:00.000Z",
  "datasourceId": "",
  "batchId": "",
  "jobStatusCode": "InQueue",
  "parameters": [
    { "id": "Query/Option/DocumentNumber", "filterType": "Single", "value": ["1"] },
    { "id": "Query/Option/Format", "filterType": "Single", "value": ["3"] },
    { "id": "Query/Filter/Job", "filterType": "Single", "value": ["154"] }
  ]
}
```

5. Job status and results

Job status codes

A job moves through the following states during processing:

Code	Name	Description
-1	Error	Job failed. StatusDescription contains the error message.
0	InQueue	Waiting to be processed.
1	Pending	Created, not yet queued.
2	Processing	Active processing started.
3	PreparingData	Fetching data from e.net.
4	Cancelled	Job was cancelled.
5	DataPrepared	Data retrieval complete.
6	ConfiguringDatasource	Setting up Wyn datasource.
7	UploadedToReportEngine	Template uploaded to Wyn.
8	ProcessingExtensions	Running post-processing extensions.
9	ExportingToPdf	Exporting to PDF.
10	PdfGenerated	PDF ready.
11	Archiving	Writing to Syspro archive.
12	Archived	Archive complete.
13	Email	Sending email.
14	Exporting	Exporting to non-PDF format.
15	Printing	Sending to printer.
16	Completed	All processing complete.

Query job status

```
GET /api/Job/{jobId}/DetailedStatus
```

Returns the current status code, status description, and job metadata.

Retrieve a completed report

When a job reaches status 16 (Completed) and the ExportReportExtension was enabled, the rendered output is stored and retrievable. Every completed job always produces a PDF copy, with no need to request it. If the job also specifies an export Format (for example Excel or Word), that format is generated in addition to the PDF, not instead of it - so the retrievable output can include more than one file: the PDF plus whichever format, if any, you asked for.

```
GET /api/Job/Download/{jobId}/{exportFormat}
```

exportFormat must match the format specified in the ExportReportExtension (e.g. PDF, Excel, Word). Returns the binary file as a download with the appropriate content type.

Preview in the browser

The process endpoint response includes a previewUrl. Append it to the base API URL and open in any browser.

```
// Response from POST /api/Report/{reportName}/ProcessReport:
{
  "jobId": "...",
  "previewUrl": "/ui/report/preview?id=..."
}

// To open: {baseUrl}{previewUrl}
```

6. Extensions

Extensions are post-processing steps that run after a report is rendered. Configure them by including extension parameters in the job request alongside the report parameters. All extension parameter IDs are prefixed with Extensions/.

To discover available extension parameters:

```
GET /api/Report/ExtensionParameters
```

Archive extension

Archives the completed report output into the Syspro archive store.

```
{ "id": "Extensions/ArchiveExtension", "filterType": "Single", "value": ["true"] }
```

For documents, the metaData field in the job request is passed through to the archive record.

Export extension

Exports the report to a specific file format and optionally writes it to a filesystem path. Also stores the output in the CompletedReports table, making it retrievable via GET /api/Archive.

```
{ "id": "Extensions/ExportReportExtension",          "filterType": "Single", "value": ["true"] },
{ "id": "Extensions/ExportReportExtension/Format",   "filterType": "Single", "value": ["PDF"] },
{ "id": "Extensions/ExportReportExtension/AddTimestamp", "filterType": "Single", "value": ["true"] },
{ "id": "Extensions/ExportReportExtension/Directory", "filterType": "Single", "value": ["C:\\Reports"] },
{ "id": "Extensions/ExportReportExtension/FileName",  "filterType": "Single", "value": [{"job.Description"}] }
```

Supported export formats

Format value	Output
PDF	Adobe PDF
Excel	Microsoft Excel (.xlsx)
Word	Microsoft Word (.docx)
CSV	Comma-separated values
HTML	HTML document
Image	PNG image
JSON	JSON data
TXT	Plain text
XML	XML data

Format value	Output
CSV Raw	CSV without formatting
Excel Data	Excel data-only (no chart formatting)

Email extension

Emails the completed report as an attachment.

```
{ "id": "Extensions/EmailExtension",          "filterType": "Single", "value":
["true"] },
{ "id": "Extensions/EmailExtension/To",      "filterType": "Single", "value":
["user@example.com"] },
{ "id": "Extensions/EmailExtension/Subject", "filterType": "Single", "value":
["Report: {job.Description}"] },
{ "id": "Extensions/EmailExtension/Body",    "filterType": "Single", "value":
["Please find the report attached."] },
{ "id": "Extensions/EmailExtension/AttachmentFormat", "filterType": "Single", "value":
["PDF"] }
```

To, Cc, and Bcc accept comma- or semicolon-separated addresses. Subject and Body support {job.Description}, {job.JobId}, and {job.OperatorCode} placeholders.

Print extension

Sends the completed PDF to a physical or virtual printer on the server.

```
{ "id": "Extensions/PrintReportExtension",    "filterType": "Single", "value":
["true"] },
{ "id": "Extensions/PrintReportExtension/PrinterName", "filterType": "Single", "value":
["Microsoft Print to PDF"] }
```

Get available printers:

```
GET /api/Printer
```

Get printer-specific settings (paper size, colour, duplex, etc.):

```
GET /api/Printer/Settings?Name={printerName}
```

Web UI notification extension

Sends a push notification to the Syspro Avanti web UI when the job completes, prompting the report viewer to refresh.

```
{ "id": "Extensions/SendWebUiNotification", "filterType": "Single", "value": ["true"] }
```

7. Scheduling

SRS includes a Quartz.NET-backed scheduler. Scheduled tasks fire at their configured time and automatically enqueue a report job. Schedule state persists across service restarts.

Create or update a scheduled task

POST /api/Scheduler/Tasks

```
{
  "taskKey": "monthly-bank-balances",
  "group": "Finance",
  "template": "list_of_bank_balances",
  "templateType": "Report",
  "description": "Monthly bank balances report",
  "parameters": [
    { "id": "Query/Option/Month", "filterType": "Single", "value": ["1"] }
  ],
  "triggers": {
    "type": "Monthly",
    "dayOfMonth": 1,
    "timeOfDay": "06:00",
    "inTimeZone": "South Africa Standard Time",
    "startDate": "2026-08-01T00:00:00Z"
  }
}
```

taskKey uniquely identifies the task within a group. Posting with an existing taskKey updates the schedule. A task cannot overwrite another user's task with the same key.

Trigger types

Type	Description
Seconds	Fire every N seconds.
Minutes	Fire every N minutes.
Hours	Fire every N hours.
Days	Fire every N days.
Weeks	Fire every N weeks on specified days.
Monthly	Fire on a specific day of the month.
Yearly	Fire on a specific date each year.

Additional scheduler endpoints

Endpoint	Description
GET /api/Scheduler/Tasks	List all tasks with state, next fire times, and cron expression.
POST /api/Scheduler/Triggers/Simulate	Preview next N fire times for a trigger definition without saving.

Endpoint	Description
POST /api/Scheduler/Tasks/Pause	Pause a scheduled task.
POST /api/Scheduler/Tasks/Resume	Resume a paused task.

8. Adding custom reports

External developers can register new third-party reports in SRS. A report requires two components: a control file (metadata) and a template (.rdlx). The report must also be added to a menu before it appears in the report browser.

This section applies to reports only. Document/stationary templates are managed through Syspro document management tools.

Step 1: Create the report control file

The control file registers the report in Syspro's file system, defining the report name, type, and associated business object.

```
POST /api/Report/{reportName}/ReportControl
```

The report ID is derived from the name you supply: lowercase, spaces replaced with underscores. For example, "List Of Logged-In Operators" becomes "list_of_logged-in_operators".

 The report ID is permanent once set. Choose the report name carefully before registering it.

Step 2: Add the report to a menu

A report must be added to the SRS menu structure before it is visible to users in the report browser.

```
// Find the folder/module ID:
GET /api/Menu

// Add the report:
POST /api/Menu
```

Example - GET /api/Menu response (excerpt, centered on the module and report you are looking for - the real response nests every module, submodule, and report this way):

```
{
  "reportMenu": {
    "modules": [
      {
        "id": "Z1",
        "description": "Administration",
        "reports": [
          { "id": "operator_amendment_journal", "description": "Operator Amendment Journal"
        }
        // ... other reports in this module omitted
      ]
      // ... other modules omitted
    ]
  }
}
```

Having found "Z1" (Administration) as the right module, add the report there. Example - adding a report to the menu:

```
{
```

```

    "name": "list_of_logged-in_operators",
    "description": "Lists all operators currently logged in to SRS",
    "module": "Z1",
    "reportType": "ThirdPartyReport"
  }

```

Use the same id for reportName in Step 1 and name in Step 2. Step 1 uses the route value as-is; Step 2 only lowercases name and keeps spaces, so a human-readable title like "List Of Logged-In Operators" produces a menu id that won't match the id registered in Step 1. Pass the lowercased, underscore-separated id as name, and put the readable title in description instead.

module must also match an existing id in the GET /api/Menu response - search the full module and submodule tree, not just the top level, the way "Z1" was found above. A module id that doesn't exist fails the request outright rather than returning a validation error, so confirm it first rather than guessing a code. The 200 response's reportId field returns a confirmation message, not the id itself - call GET /api/Menu again if you need to confirm the final id.

Step 3: Design the template

The Wyn template designer is embedded in SRS at /ui/report/design. Use the API endpoint to get the correct URL:

POST /api/Report/design-url

```

Request:
{
  "reportId": "list_of_logged-in_operators",
  "templateType": "Report",
  "templateFile": "",
  "dataSourceId": "",
  "newTemplate": true
}

Response:
{
  "designTimeUrl": "https://{gateway}/syspro/srs/ui/report/design?
reportid=...&access_token=...",
  "template": "",
  "dataSourceId": ""
}

```

Open the returned URL in a browser. For a new report (newTemplate: true), a blank template is created, and the visual designer opens immediately. Design the layout, data bindings, and formatting, then save. SRS writes the template back to Syspro storage automatically.

The designer session uses a periodic heartbeat to stay alive. If the browser tab is closed without saving, the session cleans up automatically within a few minutes.

Designer URL parameters

Parameter	Description
reportId	Report ID (lowercase_underscore).
templateType	"Report" for standard reports.
templateFile	Existing template filename for updates. Empty string

Parameter	Description
	for new reports.
dataSourceId	Pre-existing datasource GUID. Leave empty for standard reports.
newTemplate	"true" creates a blank template. "false" loads an existing template for editing.

9. Error handling

All error responses follow the RFC 7807 Problem Details format with content type `application/problem+json`.

```
{
  "type": "KeyNotFoundException",
  "title": "KeyNotFoundException",
  "status": 400,
  "detail": "Report template not found: list_of_bank_balances",
  "traceId": "00-abc123...",
  "fullErrorType": "System.Collections.Generic.KeyNotFoundException"
}
```

HTTP status codes

Status	Condition
200 OK	Request succeeded.
400 Bad Request	Invalid input: malformed JSON, duplicate name, or key not found.
401 Unauthorized	Missing, invalid, or expired bearer token.
503 Service Unavailable	Downstream failure: e.net SOAP error, Wyn engine unreachable, or unexpected server exception.

Job-level errors

Background jobs do not throw HTTP errors. They update the job record with `StatusCode = -1` and write a message to `StatusDescription`. Always check job status when polling.

```
{
  "jobId": "...",
  "statusCode": -1,
  "statusDescription": "Error: Report template not found"
}
```

10. Swagger / OpenAPI

Interactive API documentation is available at:

```
https://{gateway-host}/syspro/srs/swagger
```

The Swagger UI documents all endpoints, request/response schemas, and authentication. Use it to explore and test calls directly from your browser with your bearer token. It is also accessible from the SRS menu in the Syspro portal.

Appendix A: Quick-start checklist

Follow these steps to make your first successful report request:

- Confirm the gateway base URL with your Syspro system administrator.
- Obtain a Syspro e.net session token via the standard Syspro logon.
- POST the e.net token to `/api/Authenticate/LogonWithSysproEnetToken`. Save the returned `sessionToken`.
- GET `/api/Report/AvailableReports` to find the report ID you want to run.
- GET `/api/Report/{reportName}/Parameters` to understand required input.
- POST `/api/Report/{reportName}/ProcessReport` with the job payload. Include the `ExportReportExtension` if you need to retrieve the output file.
- Check the response for `jobId` and `previewUrl`.
- If you submitted to the queue instead: poll GET `/api/Job/{jobId}/DetailedStatus` until `statusCode = 16`.
- Retrieve the exported file via GET `/api/Job/Download/{jobId}/{exportFormat}`.

Appendix B: Glossary

Term	Definition
SRS	Syspro Reporting Services — the .NET 8 Windows Service described in this guide.
Wyn	Third-party report rendering engine embedded in SRS. Handles template rendering and export.
e.net	Syspro SOAP/WCF business object layer. SRS calls it for data, templates, auth, and archiving.
Job	A single report processing request, tracked from queue to completion via a job ID.
Template	An .rdlx file defining report layout and data bindings, stored in Syspro and uploaded to Wyn at processing time.
Extension	A post-processing step (archive, export, email, print, notification) configured as part of a job request.
Application Gateway	Reverse proxy managed by the Syspro platform. All external API traffic passes through it.
BO	Business Object — a Syspro e.net function that retrieves or manipulates ERP data.
datasourceId	Wyn datasource identifier. Leave empty for standard jobs; SRS manages datasource lifecycle automatically.

syspro