

ADMINISTRATION GUIDE

Synergi Pipeline 8

Assessment Results Service Application Programming Interface (API)

DECEMBER 2022
DNV - Digital Solutions

About DNV

We are the independent expert in risk management and quality assurance. Driven by our purpose, to safeguard life, property, and the environment, we empower our customers and their stakeholders with facts and reliable insights so that critical decisions can be made with confidence. As a trusted voice for many of the world's most successful organizations, we use our knowledge to advance safety and performance, set industry benchmarks, and inspire and invent solutions to tackle global transformations.

Digital Solutions

DNV is a world-leading provider of digital solutions and software applications with focus on the energy, maritime and healthcare markets. Our solutions are used worldwide to manage risk and performance for wind turbines, electric grids, pipelines, processing plants, offshore structures, ships, and more. Supported by our domain knowledge and Veracity assurance platform, we enable companies to digitize and manage business critical activities in a sustainable, cost-efficient, safe, and secure way.

Edition

This edition applies to Synergi Pipeline 8 and to subsequent releases and modifications until otherwise indicated in new editions.

Trademarks

All brands and product names are trademarks of the respective owner.

Restricted rights, warranties, and liabilities

The software discussed in this document is provided under a Software License Agreement and may be used or copied only in accordance with the terms of that license. All warranties given by DNV AS concerning DNV asset software are set forth in the Software License Agreement between DNV AS and the licensee.

DNV assumes no responsibility for any errors that may appear in this document. We reserve the right to change our software and documentation without notice.

Use, duplication, or disclosure by the U.S. Government is subject to the restrictions defined as "Rights specified in the license" as set forth in subdivisions (a) and (b) of the DFARS clause 227.7202-3 entitled Rights in Commercial Computer Software and Commercial Computer Software Documentation.

Copyright notice

© 2022 DNV AS.
www.dnv.com/digital

Copyrights

Python Slugify

The Synergi Pipeline application uses the Python Slugify project. Python Slugify uses the “text-unidecode” module under the Artistic License 2.0. You may obtain a copy of the original source code of text-unidecode at: <https://github.com/kmike/text-unidecode>.

Table of Contents

SYNERGI PIPELINE ASSESSMENT RESULTS SERVICE API	1
Overview	1
Design	2
REST POST and GET Calling Patterns	3
Preparation	3
Collection	3
Security	4
Deployment	5
Deploying the Assessment Results Service API	5
Configuration Settings	8
Settings controlling use of Mock/Dummy data	8
Response compression settings	9
Security Settings - Startup	9
Security Settings - Other	10
Service job and cache storage settings	10
Database Connection Strings	11
Connection strings for the results service state cache tables	11
Connection strings for the results service reading Synergi Pipeline results and metadata	11
System Operational Options	12
Logging	16
AssessmentResultsService Controller – API Workflow Example	17
HealthMonitor Controller – API Workflow Example	19
Cleanup Controller – API Workflow Example	19
APPENDIX A: MOCK MODE/DUMMY DATA GENERATION	21
FOR MORE INFORMATION	30

Synergi Pipeline Assessment Results Service API

This Administration Guide provides information for the configuration, and implementation of the Synergi Pipeline Assessment Results Service Application Programming Interface (API). The Assessment Results Service API provides a secure mechanism that makes approved Risk, Analytical, HCA and MCA, and Class Location assessment results available to users who do not have direct access to the reporting database.

The SpiAssessmentResultsService component is a .NET Core microservice that connects and reads results data from Synergi Pipeline's product and reporting databases. It prepares calculational results for download using a REST API.

Instructions for on-premise deployment of the Synergi Pipeline Assessment Results Service API are provided in "[Deployment](#)" on page 5 .

For Synergi Pipeline Software as a Service (SaaS) deployments, DNV implements the Synergi Pipeline Assessment Results Service API deployment process for your Synergi Pipeline environment.

For further information about the Synergi Pipeline Assessment Results Service API or to request assistance, contact your Solution Support Manager or DNV - Digital Solutions Technical Support at software.support@dnv.com.

Overview

The Assessment Results Service API does not require Synergi Pipeline software components to operate. It only requires a connection to a known version of the Synergi Pipeline database(s). The service itself allows for synchronous preparation and asynchronous collection of approved results from Synergi Pipeline's database. The service manages the state of this work through three database tables it creates, which can be resident in either of the four Synergi Pipeline databases or put in an entirely new separate database – including a local SQLite file database – if needed. The tables managed by this service maintain the state of results requests (jobs), which can be collected asynchronously and retained for later download by API consumers.

The service requires three support tables and maintains automated migrations of the three tables during startups. These tables are:

- **AR_JOBS** – Holds one record for each result/job request initiated by consumers of the API.
- **AR_PAYLOADS** – Holds one record for each result/job and payload combination. This is the cache registry of all collected payloads.

- **AR_KEYVALUEPAIRS** – Holds one record for each result/job and payload combination in blocks. Records in this table hold the actual compressed “Collected” results in ready-made export format. Data in this table is in JSON format but compressed. This table is likely to see the most growth depending upon number of jobs, payloads, size of results, and the retention periods desired between the collection and the retrieval of the output payloads.

NOTE: Several administration methods found on the Cleanup controller allow administrators to clean up cached results job information.

Design

The service is designed to scale and perform well given large and complex sets of results resident across several tables in the Synergi Pipeline database. It supports expansion of additional results types to be requested (provided in the future beyond Risk, Analytical, HCA and MCA, and Class Location assessment results), and it allows additional output types as well (provided in the future beyond the initial offering of downloads of JSON payloads). It runs in both Azure and on-premise environments. The service can retrieve large results sets, which take many minutes to generate without having callers wait or timeout while results are being collected. It also provides download of results in manageable payload sizes (which may be too large to send back on a single REST response message).

Due to the potential time necessary to collect some results sets (greater than four minutes in many cases), and also the size of the return record set(s) from the results (greater than one GB and several million records in some cases) – it provides a REST asynchronous pattern. The service contains a default endpoint, which is fully self-documented using online Swagger API documentation and examples. It also contains several API endpoints to support administration (Health Monitor) and management of its results state cache (Cleanup).

The service supports three controllers and various REST API endpoints on each:

1. **AssessmentResultsService Controller** — This is the main controller that consumers of the API will interact with to perform results extractions. Its base end-point pattern is: {domain}/api/AssessmentResultsService (For further information, see ["AssessmentResultsService Controller – API Workflow Example"](#) on page 17.)
2. **HealthMonitor Controller** — This is a health-check controller that can be used to ping the service to gain useful life and configuration information. Its base end-point pattern is: {domain}/api/HealthMonitor (For further information see ["HealthMonitor Controller – API Workflow Example"](#) on page 19.)
3. **Cleanup Controller** — This is an administration controller that can be used to check and clean up the results/job storage cache. Its base end-point pattern is: {domain}/api/Cleanup (For further information, see ["Cleanup Controller – API Workflow Example"](#) on page 19.)

REST POST and GET Calling Patterns

The main `AssessmentResultsService` controller supports a single REST pattern is used in all cases to retrieve both large and small results. A minimum of two REST calls are necessary: a "POST" call and at least one or more "GET" calls.

Preparation

The results are requested with a REST POST call with input parameters; the input parameters are contained on a JSON payload called the `resultsSpecificationDTO`. The `resultsSpecificationDTO` specifies the type of results set being requested and all other user-allowed input options. The call is validated immediately, and some summary statistics about the request are gathered. If valid, it is synchronously returned in under four minutes using an output JSON payload called the `ResultsReadyDTO`. The `ResultsReadyDTO` indicates whether the job's collection step is completed or currently running (that is, either `resultsReadyDTO.resultsReady = true`, or `resultsReadyDTO.resultsReady = false`). This step is called "Preparation".

Collection

In parallel with returning the results of the "Preparation" step, the service also spawns a background task internally to begin the results "Collection" step. Results collection is the process of gathering the results (and preparing them into the format) and payloads for future return to the user. Results are not marked as ready for user download until all payloads are generated and cached into the service's results state cache.

When results are ready, results metadata is returned, with validation information and the status when the user polls the service. When ready, the results can be retrieved immediately on subsequent "GET" calls with the information provided in the `ResultsReadyDTO` response.

Polling calls (comprised of 0-N) GETs are required by the consumer until the service's "Collect" step is completed.

The results will continue to be constructed on the server, but the client will receive (`resultsReadyDTO.resultsReady=false`) and will re-poll periodically by using the "resultId" for the request until `resultsReadyDTO.resultsReady = true`.

When final results collection is complete (`resultsReadyDTO.resultsReady=true`), the response will contain the one or more URL addresses to be used to get/pull the single or multiple payload/packages for the result. One or more similar "GET" calls will be used to pull each payload/package as needed.

Security

Security is provided by requiring recognized bearer tokens on all service calls. The tokens require an encrypted secret, expiration date/time, and a "READ" user role. Bearer tokens can be generated by the service itself (if configured to allow this) given a set of credentials. Alternatively, they can also be generated by the client consumer (if allowed) by administrators through means of a shared secret configured in the service. Bearer tokens require this secret and also an expiration date/time and a "READ" user role, which are all configurable.

Inside the service there are three stages of fulfilling a results request:

- **Get or generate bearer token**
 - These can be self-generated given knowledge of the public "secret" resident in the service or can be generated by the service (using REST call `generatetoken`) given credentials configured in the service.
 - Valid bearer tokens are then attached to the header of all requests.
- **Preparation** - Returned synchronously
 - Post `ResultsRequest` call using `resultsSpecificationDTO`. Preparation will perform request validation and sizing and will quickly return a result (less than four minutes) using the `ResultsReadyDTO`.
 - Returns `ResultsReadDTO.resultsReady = false`
 - Also initiates the results "Collection" step in the background (if the request is valid).
- **Collection** – Returned asynchronously; must be polled until completion.
 - Results collection can take 0-n minutes depending on volume and complexity of the results; users are provided a "resultId" for use in subsequent polling requests.
 - Returns `ResultsReadyDTO.ResultId = (guid)` containing the resultID for future polling and payload extractions.
 - Returns `ResultsReadyDTO.resultsReady = true` when ready; false indicates that the collect step is still running.
 - `ResultsReadyDTO.resultsPayloadAddresses (1-n)` – may entail one or more payload retrievals, sized by record count in configuration.

In addition to normal operations (connected to a Synergi Pipeline database), the service also offers a "mock" (or dummy) operational mode, where it will return mock example data without being connected to any DBMS. This mode can be used for trials or API exploration without need of a Synergi Pipeline database with real data. Its use is described more fully in ["Appendix A: Mock Mode/Dummy Data Generation"](#) on page 21.

Deployment

Instructions for on-premise deployment of the Synergi Pipeline Assessment Results Service API are provided in this section.

For Synergi Pipeline Software as a Service (SaaS) deployments, DNV implements the Synergi Pipeline Assessment Results Service API deployment process for your Synergi Pipeline environment.

For further information or assistance, contact the direct technical support contact person listed in your contract agreement or contact DNV - Digital Solutions Technical Support at software.support@dnv.com.

Deploying the Assessment Results Service API

For on-premise deployments of the Synergi Pipeline Assessment Results Service API, the on-premise installation script looks for a copy of the appsettings.json file in the Setup folder and copies it into place after deployment.

1. Configure the appsettings.json file and name it: ARS_appsettings.json
2. Copy it to the wwwroot\sitename\SETUP folder where it will persist across installs.
3. Run an install so the powershell will move it into place.

NOTE: You can optionally copy it manually into the service as follows:

folder: \PIMSWeb\NetCoreServices\DNVGL.Spi.AssessmentResultsService.Api

file name: appsettings.json

An example of configuration for the appsettings.json file is as follows:

```
{
  "Startup": {
    "IsMock": "False",
    "MockPayloadRecordCount": 12,
    "MockTotalRecordCount": 12,
    "IsCORS": "False",
    "UseGZipCompression": "True", //true = GZip on, false = no compression
    "GZipCompressionType": "Optimal", //"Optimal", "Fastest", "NoCompression"
    "AllowBearerTokenGeneration": "True", //callers can ask the service to generate their own tokens
```

```
"GeneratedBearerTokenUserName": "MyUser", //callers user name to check if generating tokens
"GeneratedBearerTokenPassword": "MyPassword", //callers password name to check if generating
tokens
"GeneratedBearerTokenExpHrs": 0, //generated tokens expiration period
"GeneratedBearerTokenExpMins": 10, //default 10 minutes
"GeneratedBearerTokenExpSecs": 0
},
"CacheStorage": {
  //"Type": "SqlLite"
  "Type": "MSSQL"
},
"JobsStorage": {
  "Type": "MSSQL"
  //"Type": "SqlLite"
},
"JobHandler": {
  "CollectAsynchronously": true,
  "CacheRetentionTimeHrs": 720, //1 month
  "CleanupOnStartup": true,
  "CleanupTimeoutMins": 10
},
"ConnectionStrings": {
  "ConnectionType": "SqlServer",
  "JobsStorageConnection": "server=SQLServerName;database=Database_Name_
REP;uid=orbitpipeline;pwd=SQLPassword;", //typically the spi rep database, but can be any
  "CacheStorageConnection": "server=SQLServerName;database=Database_Name_
REP;uid=orbitpipeline;pwd=SQLPassword;", //typically the spi rep database, but can be any
  "ConnectionPM": "server=SQLServerName;database=TEST_SP_TRANS_SI_8_6_0_0_
PM;uid=orbitpipeline;pwd=SQLPassword;", //spi pm database
```

```
"ConnectionRep": "server=SQLServerName;database=Database_Name_
REP;uid=orbitpipeline;pwd=SQLPassword;" //spi rep database
},
//internal JWT is meant to be the secret from SPi
"Jwt": {
  "Secret": "SecretGUID"
},
//publish JWT is meant to be different from internal, and for use by outside API consumers
"PublicJwt": {
  "Secret": "SecretGUID2"
},
"Services": {
},
"SystemOptions": {
  "CacheBlockRecordCount": 10000,
  "DatabaseFetchSize": 50000,
  "AdminGuideFolderName": "Documents", //name of admin guide folder on server disk
  "AdminGuideFileNamePDF": "SynPipelineAssessResultsSvcAPI_AdminGuide.pdf", //name of
admin guide in service footprint so user can download if desired
  "AdminGuideFileNameHTML": "AdminGuide.html" //name of admin guide in service footprint so
user can display if desired
},
"Logging": {
  "LogLevel": {
    "Default": "Information"
  }
},
"AllowedHosts": "*"
}
```

Configuration Settings

This section describes configuration settings, including:

- [Use of Mock/Dummy Data](#)
- [Response Compression](#)
- [Security](#)
 - [Startup](#)
 - [Other](#)
- [Service Job and Cache Storage](#)
- [Database Connection Strings](#)
 - [Results service state cache tables](#)
 - [Results service reading Synergi Pipeline results and metadata](#)
- [System Operational Options](#)
- [Logging](#)

Settings controlling use of Mock/Dummy data

"Startup": "IsMock": "False", (string, True/False)	If False, the connectionSettings options below will connect the service to one or more DBMSs. If True, the service will not be connected to any DBMS, but will respond with mock/dummy data to all requests. IsMock = True is used for testing and debugging purposes only.
"Startup": "MockPayloadRecordCount": 12, (Integer 1-N)	Number of records per payload, by default, generated when using IsMock=True
"Startup": "MockTotalRecordCount": 12, (Integer 1-N)	Total number of records generated, by default, when using IsMock=True

Response compression settings

"Startup": "UseGZipCompression": "True", (string, True/False)	If True = GZip compression will be on by default for all returned REST responses. If False = No compression of REST responses.
"Startup": "GZipCompressionType": "Optimal" (string, keyword: "Optimal", "Fastest", "NoCompression")	Controls the type of Gzip compression provided by the service on its responses.

Security Settings - Startup

"Startup": "IsCORS": "False", (string, True/False)	Allows cross-site request calls true or false. In production this should be = False to ensure proper bearer token validation and to prevent cross-site-scripting attacks.
"Startup": "AllowBearerTokenGeneration": "True", (string, True/False)	Indicates whether the API will support a call from users to /generatetoken to generate their own bearer tokens with input credentials. If True the /generatetoken endpoint will be active. If False, it will be disabled, and users must generate their own tokens externally.
"Startup": "GeneratedBearerTokenUserName": "MyUser", (string)	If AllowBearerTokenGeneration = True, desired name credential user may provide through the generatetoken API.
"Startup": "GeneratedBearerTokenPassword": "someHardToGuessPasswordGoesHere", (string)	If AllowBearerTokenGeneration = True, desired password credential user may provide through the generatetoken API.
"Startup": "GeneratedBearerTokenExpHrs": 0, (integer, 0-N)	If AllowBearerTokenGeneration = True, desired default number of hours for tokens generated through the generatetoken API. If users specify this number in their incoming request, it will override this default.

<p>"Startup": "GeneratedBearerTokenExpMins": 10, (integer, 0-N)</p>	<p>If AllowBearerTokenGeneration = True, desired default number of minutes for tokens generated through the generatetoken API. If users specify this number in their incoming request, it will override this default.</p>
---	---

<p>"Startup": "GeneratedBearerTokenExpSecs": 0 (integer, 0-N)</p>	<p>If AllowBearerTokenGeneration = True, desired default number of seconds for tokens generated through the generatetoken API. If users specify this number in their incoming request, it will override this default.</p>
---	---

Security Settings - Other

<p>"InternalJwt": "Secret": "someEncryptedStringHere" (string)</p>	<p>Contains Synergi Pipeline's encrypted product secret. For use in communications between Synergi Pipeline's other services internally.</p>
--	--

<p>"PublicJwt": "Secret": " someOtherEncryptedStringHere " (string)</p>	<p>Contains Synergi Pipeline's encrypted external product secret. For use in communications between consumers of the public API externally. If AllowBearerTokenGeneration = False, then this secret would have to be shared with external consumers of the results API so that they can generate their own valid bearer tokens externally.</p>
---	--

<p>"AllowedHosts": "*" (string)</p>	<p>Used for host filtering to optionally bind this application to specific host names. For example, if you replace "*" with "example.com", then the service will be restricted to run only under: http://example.com/</p>
-------------------------------------	---

Service job and cache storage settings

<p>"CacheStorage": "Type": "MSSQL" (string, keyword: "MSSQL", "SqlLite", "Oracle")</p>	<p>Type of DBMS storage for the service's Payload Cache tables (AR_KEYVALUEPARS, AR_PAYLOADS)</p>
--	---

<p>"JobsStorage": "Type": "MSSQL" (string, keyword: "MSSQL", "SqlLite", "Oracle")</p>	<p>Type of DBMS storage for the service's Jobs Cache table (AR_JOBS)</p>
---	--

<code>"JobHandler":</code> <code>"CollectAsynchronously":</code> <code>true (Boolean True/False)</code>	Controls whether the "collect" step will be Synchronous (false) or Asynchronous (true). In normal operation in production, this value should be set to "true" so that the collection step (which might take several minutes) will be run in the background and not attached to the "Prepare" step synchronously.
---	--

Database Connection Strings

Connection strings for the results service state cache tables

`"ConnectionStrings": "JobsStorageConnection": "sqlServer or Oracle connection String to the Jobs storage (AR_Jobs)",`

`"ConnectionStrings": "CacheStorageConnection": "sqlServer or Oracle connection String to the results cache storage (AR_Payloads, AR_KeyvaluePairs)",`

SQLite Example: `"ConnectionStrings": "JobsStorageConnection": "FileName=C:\\temp\\SpiAssessmentResultsJobs.db",` - Example connection string containing drive, path, and filename, if JobsStorage, or CacheStorage. Type = 'SQLite'.

NOTE: SQLite is normally used for localized testing and debugging, not for production deployments due to possible disk space constraints.

Connection strings for the results service reading Synergi Pipeline results and metadata

`"ConnectionStrings": "ConnectionType": "SqlServer", (string, keywords: "SqlServer", "Oracle")`

`"ConnectionStrings": "ConnectionPM": "sqlServer or Oracle connection string to the Synergi Pipeline Product (PM) database – metadata is retrieved from here",`

`"ConnectionStrings": "ConnectionRep": "sqlServer or Oracle connection string to the Synergi Pipeline Reporting (REP) database – results data is retrieved from here"`

System Operational Options

<code>"SystemOptions": "CacheBlockRecordCount": 10000, (integer, 1-N)</code>	Controls the size of payloads (blocks) constructed during the "Collect" phase of processing. This number controls the number of records compressed onto each block in internal cache storage in the AR_Keyvaluepairs cache table as well as the number of records returned per payload for each results request. For example, if the overall results set contains 100,000 records and this value is set to 10,000, then the services collect processing will construct 10 payloads of 10,000 records each to be made ready for user download. This number should be tuned based on desired consumer output and networking constraints. It controls the record count size of payloads users wish to download from the server.
<code>"SystemOptions": "DatabaseFetchSize": 50000, (integer, 1-N)</code>	Controls the number of records fetched per page when reading through the database. This number controls how many records at a time will be resident in the server "Collect" processing before being flushed to the results cache tables. In effect this controls the size of the in-memory record set in the service at a given time during "Collect". This value should be tuned based on DBMS access and also on peak memory loads available to the service.
<code>"SystemOptions": "CleanupOnStartup": "False", (string, True/False)</code>	Controls whether or not cache clean up occurs upon start-up. Enabling this option helps keep the cache from growing too large, which can cause timeouts to occur during start-ups.

"RiskAssessmentEngine":
"TransmissionCacheBlockRecordCount":
100, (integer, 1-N)

Optional setting to control and tune the cache block record count for transmission risk models, superseding the default "SystemOptions.CacheBlockRecordCount" value. Transmission data is usually shaped very differently to distribution data, having far fewer analysis items but with much more data on them. The default "SystemOptions.CacheBlockRecordCount" value of 10,000 could therefore consume too much memory in the service to store all the analysis items.

"RiskAssessmentEngine":
"TransmissionDatabaseFetchSize": 100,
(integer, 1-N)

Optional setting to control the number of records fetched per page when reading through the database for transmission risk models, superseding the default "SystemOptions.DatabaseFetchSize" value. Transmission data is usually shaped very differently to distribution data, having far fewer analysis items but with much more data on them. The default "SystemOptions.DatabaseFetchSize" value of 10,000 could therefore return too much data for the service to process and unnecessarily tax the database server.

"RiskAssessmentEngine":
"DistributionCacheBlockRecordCount":
10000, (integer, 1-N)

Optional setting to control and tune the cache block record count for distribution risk models, superseding the default "SystemOptions.CacheBlockRecordCount" value. See "SystemOptions.CacheBlockRecordCount" for more details.

"RiskAssessmentEngine":
"DistributionDatabaseFetchSize": 50000,
(integer, 1-N)

Optional setting to control the number of records fetched per page when reading through the database for distribution risk models, superseding the default "SystemOptions.DatabaseFetchSize" value. See "SystemOptions.DatabaseFetchSize" for more details.

"AnalyticalAssessmentEngine":
"TransmissionCacheBlockRecordCount":
100, (integer, 1-N)

Optional setting to control and tune the cache block record count for transmission analytical models, superseding the default "SystemOptions.CacheBlockRecordCount" value. Transmission data is usually shaped very differently to distribution data, having far fewer analysis items but with much more data on them. The default "SystemOptions.CacheBlockRecordCount" value of 10,000 could therefore consume too much memory in the service to store all the analysis items.

"AnalyticalAssessmentEngine":
"TransmissionDatabaseFetchSize": 100,
(integer, 1-N)

Optional setting to control the number of records fetched per page when reading through the database for transmission analytical models, superseding the default "SystemOptions.DatabaseFetchSize" value. Transmission data is usually shaped very differently to distribution data, having far fewer analysis items but with much more data on them. The default "SystemOptions.DatabaseFetchSize" value of 10,000 could therefore return too much data for the service to process and unnecessarily tax the database server.

"AnalyticalAssessmentEngine":
"DistributionCacheBlockRecordCount":
10000, (integer, 1-N)

Optional setting to control and tune the cache block record count for distribution analytical models, superseding the default "SystemOptions.CacheBlockRecordCount" value. See "SystemOptions.CacheBlockRecordCount" for more details.

"AnalyticalAssessmentEngine":
"DistributionDatabaseFetchSize": 50000,
(integer, 1-N)

Optional setting to control the number of records fetched per page when reading through the database for distribution analytical models, superseding the default "SystemOptions.DatabaseFetchSize" value. See "SystemOptions.DatabaseFetchSize" for more details.

"ClassLocationAssessmentEngine": { "CacheBlockRecordCount": 10000 }, (integer, 1-N)	Optional setting to control and tune the cache block record count for class location, superseding the default "SystemOptions.CacheBlockRecordCount" value. See "SystemOptions.CacheBlockRecordCount" for more details.
"HcaAssessmentEngine": { "CacheBlockRecordCount": 10000 }, (integer, 1-N)	Optional setting to control and tune the cache block record count for HCA and MCA, superseding the default "SystemOptions.CacheBlockRecordCount" value. See "SystemOptions.CacheBlockRecordCount" for more details.
"SystemOptions": "CleanupTimeoutMins":10, (integer, 1-N)	Controls the length of time during which cache clean up can occur before the start-up process times out. Large cache clean-ups can cause start-ups to time out if this setting is not long enough.
"SystemOptions": "CacheRetentionTimeHrs": 5000, (integer, 1-N)	Controls the length of time jobs and cached results payloads will stay in the service's state cache tables awaiting retrieval. Jobs and cached ("collected") payloads older than this value will become candidates for expiration/removal by the service.
"SystemOptions": "AdminGuideFolderName": "Documents", (string)	Folder name on server disk. Contains the name of the Administration Guide folder on server disk. Swagger endpoint documentation allows downloads of the server's Administration Guides and online review. This folder on disk contains the latest source documents.
"SystemOptions": "AdminGuideFileNamePDF": "AdminGuide.pdf", (string)	Name of Administration Guide in service footprint (in PDF format) so users can download if desired using the Swagger endpoint documentation.
"SystemOptions": "AdminGuideFileNameHTML": "AdminGuide.html" (string)	Name of Administration Guide in service footprint (in HTML format) so users can display it online using the Swagger endpoint documentation.

Logging

"Logging": "LogLevel": "Default": "Information" - (string, Keyword: "Trace", "Debug", "Information", "Warning", "Error", "Critical", "None")	ASP.NET core log level keywords. Controls .NET Core log level. Log amount and type of logging messages written to the log file during operation. ASP.NET Core and DNV code maintains this hierarchy of log levels. Refer to .NET Core documentation for additional logging options and settings.
--	--

log4net.config	Log persistence is further controlled through a local file on the service folder called: log4net.config. This file contains detailed directives specific to Log4Net logging used by the service to persist the log file(s) or database updates, including the type and location of all logs, log retention policies, and other options. Refer to Log4Net documentation for additional logging options controlled thru this file.
----------------	--

AssessmentResultsService Controller – API Workflow Example

A draft API workflow example is provided in this section. Also, consumers of the API should consult the online Swagger API documentation for examples of all JSON payloads.

Get or generate bearer token

- Can be self-generated given the internal "secret", or can be generated by a service call given credentials
 - Specify an expiration time on token
 - Specify the "READ" user role (for self-generated tokens; tokens generated automatically already have the "READ" user (Basic User) role assigned)
- Attach bearer token to all other POST and GET requests

POST {spi domain}/api/AssessmentResultsService/generatetoken

Post body for generatetoken request

```
{  
  "userName": "MyUser",  
  "password": "MyPassword",  
  "expMins": 5, //optional  
}
```

Response:

```
{ "token": "Bearer  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWV1aWQiOiJBc3Nlc3NtZW50UmVzdWx0  
c0FaUdlbmVYXRlZC1NeVVzZXIiLCJuYmYiOiJlMjMTY3MDEyOTAsImV4cCI6MTYxNjcwMTU5MCwiaW  
F0IjoxNjE2Nz  
AxMjkwfQ.pbyArssp69H8VI7_hMWkQGmDDPB1fT8gfNEeg93nhOM",  
  "expirationDateTime": "2021-03-25T19:46:30.3508349Z" }
```

The token value "Bearer... and so on" string would be put into the authentication header on all calls shown in the following section.

POST {spi domain}/api/AssessmentResultsService/results

Authorization: Bearer or JWT access token, or other authorization header here

Content-type: application/JSON

Post Payload = (resultsSpecificationDTO) RiskAssessmentApproved

Post Payload = (resultsSpecificationDTO) AnalyticalAssessmentApproved

Post Payload = (resultsSpecificationDTO) HcaAssessmentApproved

Post Payload = (resultsSpecificationDTO) CIAssessmentApproved

If validation fails: 400 response or other REST error code is issued, including message about what was wrong with the input.

Response = (resultsErrorDTO)

If validation succeeds (and result not ready yet): Result is not ready yet

Response = (resultsReadyDTO, resultsReady=false)

If not ready yet, client should POLL periodically until resultsReady=true (estimated polling frequency can vary, but every 60 seconds might be reasonable)

Poll with resultId for results status periodically until results generation is completed

GET {spi domain}/api/AssessmentResultsService/results/06b4bdfd-aa41-4909-b417-56d6b8d01f8e

Response = (resultsReadyDTO.resultsReady = false)

Try again, until:

Response = (resultsReadyDTO.resultsReady = true)

Using the resultsPayloadAddresses, the results data can then be returned on 1 thru N subsequent calls for each payload using resultsReadyDTO URL addresses as shown above.

ResultsPayload collection addresses are called to "GET" each payload with the resultId. The ResultsPayloadDTO will be retrieved for each payload set of records.

GET {spi domain}/api/AssessmentResultsService/results/06b4bdfd-aa41-4909-b417-56d6b8d01f8e/payload/0

Response = (ResultsPayloadDTO) ...1 of 6

GET {spi domain}/api/AssessmentResultsService/results/06b4bdfd-aa41-4909-b417-56d6b8d01f8e/payload/1

Response = (ResultsPayloadDTO) ... 2 thru 6, and so on until done and last payload retrieved

And so on...

And can be requested with second "GET" call given the resultId and URL:

GET {spi domain}/api/AssessmentResultsService/results/06b4bfdf-aa41-4909-b417-56d6b8d01f8e/payload/0

Response = (ResultsPayloadDTO) ...1

HealthMonitor Controller – API Workflow Example

The HealthMonitor controller is typically used for service debugging and diagnostic review.

GET {spi domain}/api/HealthMonitor

Response = REST Return code = 200 = OK

Service is running; any other response indicates service is not up or not responding.

GET {spi domain}/api/HealthMonitor/configuration

Response = (ServiceConfigurationDTO)

Information about the service's current configuration parameters.

GET {spi domain}/api/HealthMonitor/state

Response = (ServiceStateDTO)

Information about the service's DBMS connection state, version, and DBMS connection information for its Jobs repository.

Cleanup Controller – API Workflow Example

The Cleanup controller is typically used for service administration and management of the saved results and job caches. This controller will manage removal of selected results jobs and cached payloads, as well as any incomplete cache records that may have resulted from incomplete operations. It will take two calls to delete contents of the cache. One call will list the cache's IDs and information and the second call: /Confirmation?{use first call's parms here}

DELETE {spi domain}/api/Cleanup/All

Response = (confirmation query parms will be returned, which can be used to call the cleanup/confirmation end point to clean all job/cache information. Use the response string re-issued as the parameters in the /confirmation/ call below).

DELETE {spi domain}/api/Cleanup?resultId={myresultid here}

Response = (confirmation query parms will be returned, which can be used to call the cleanup/confirmation end point to clean a single job from the cache information. Use the response string re-issued as the parameters in the /confirmation/ call below).

DELETE {spi domain}/api/Cleanup/Obsolete

Response = (confirmation query parms will be returned, which can be used to call the cleanup/confirmation end point to clean any incomplete or obsolete (aged) cache information. Use the response string re-issued as the parameters in the /confirmation/ call below).

No obsolete job information will respond with this, meaning nothing left to do:

```
{ "confirmationUrl": "/api/results/Cleanup/confirm", "results": [], "blocks":  
[], "totalRecordsToBeDeleted": 0 }
```

DELETE {spi domain}/api/Cleanup/confirmation?{query parms used to delete single or multiple cache records}

Response = REST response code.

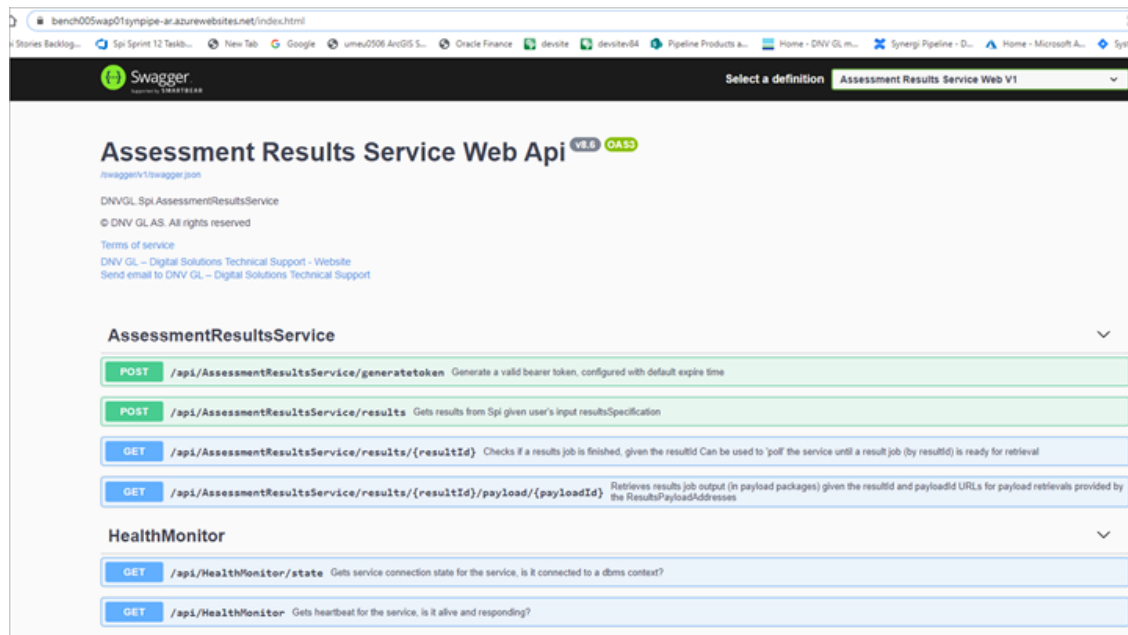
Appendix A: Mock Mode/Dummy Data Generation

This appendix describes how to use the sample API and mock/dummy data. Current outputs available in Mock form are limited to Risk and Analytical assessments.

When running with configuration setting IsMock = True, the service is not connected to any database and also has no memory of past calls or resultIds. All calls are stateless and all data generated is internal mockup data. ResultIds given are all automatically generated and are not validated between calls – you can send any ResultIds for now.

1. Getting the API documentation (Swagger endpoint)

The default service endpoint displays the Swagger documentation:



2. Generating your own bearer token

This function is controlled by settings in the appsettings configuration, and the username and password must match those configured as follows:

```
"Startup": {
  "ValidateBearerTokens": "False",
```

"AllowBearerTokenGeneration": "True", //callers can ask the service to generate their own tokens

"GeneratedBearerTokenUserName": "MyUser", //callers user name to check if generating tokens

"GeneratedBearerTokenPassword": "MyPassword", //callers password name to check if generating tokens

"GeneratedBearerTokenExpHrs": 0, //generated tokens expiration period

"GeneratedBearerTokenExpMins": 5, //default 5 minutes

"GeneratedBearerTokenExpSecs": 0

},

POST/api/AssessmentResultsService/generatetoken

Post message body:

```
{
  "userName": "MyUser",
  "password": "MyPassword",
  "expHrs": 0, //optional, if any of the three exp parms are specified this will override the
  expiration duration configured in the service
  "expMins": 30, //optional
  "expSecs": 0 //optional
}
```

Returned:

```
{
  "token": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWV1aWQiOiJBc3Nlc3NtZW50UmVz
dWx0c0FwaUdlbmVyYXRlZC1NeVVzZXIiLCJuYmYiOiJlMTQ3ODAwOTIsImV4cCI6MTYx
NDc4MjI5MiwiYW0IjoxNjE0NzgwNDkyfQ.guUmrYXfJI09dVlUgKEzE-
metXwRSi6DtxXU7HporsY",
  "expirationDateTime": "2021-03-03T14:38:14.5119531Z"
}
```

3. Requesting a result

POST/api/AssessmentResultsService/results

(ResultsSpecificationDTO)

for example: can ask for each type in the ResultsType enum

```
{
  "resultsScope": {
    "modelName": "my model name",
    "resultsType": "RiskAssessmentApproved",
    "returnFormat": "json",
  },
}
```

In reality, you will receive resultsReady=false from this first call, and you would then (poll) and recall the query API in Step 4 until it returns resultsReady=true.

For the dummy mockup, you will receive a resultsReady=true from this first call, and a payload address you can use in queries below.

```
{
  "resultsInfo": {
    "resultId": "033ed29f-5024-48d7-9cd0-5b2442eab5e5",
    "resultType": "RiskAssessmentApproved",
    "modelName": "my model name",
    "assessments": [
      {
        "id": "9b223c84-816a-4c6d-8efd-195819fd3043",
        "name": "approved assessment 1"
      },
      {
        "id": "865caedf-d799-4556-8eaa-13127a637574",
        "name": "approved assessment 2"
      }
    ]
  }
}
```

```
    ],
    "startDate": "2021-03-03T14:34:09.157949Z",
    "endDate": "2021-03-03T14:36:14.157949Z",
    "elapsedSeconds": 125,
    "totalRecordCounts": [
      {
        "key": "analysisItems",
        "value": 12
      }
    ],
    "totalPayloadCount": 1,
    "resultsReady": true
  },
  "resultsSpecification": {
    "resultsScope": {
      "modelName": "my model name",
      "resultsType": "RiskAssessmentApproved",
      "returnFormat": "json"
    },
    "resultsFormulaGroups": [],
    "resultsKeyDataProperties": []
  },
  "resultsPayloadAddresses": [
    {
      "url": "/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0-5b2442eab5e5/payload/0"
    }
  ]
}
```

4. Querying if the result is ready

When `resultsReady=false`, this method is then used for polling until the service replies that its results are ready for extraction. So this method would go into a loop (for example, every 60 seconds), until it returns with `resultsReady = true`.

GET/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0-5b2442eab5e5

Using any request ID, you can test the poll method (called as shown in the example above). It will respond that results are ready and give the payload address you can use for Step 5.

```
{
  "resultsInfo": {
    "resultId": "033ed29f-5024-48d7-9cd0-5b2442eab5e5",
    "resultType": "RiskAssessmentApproved",
    "modelName": "my risk model",
    "assessments": [
      {
        "id": "401d7feb-9d13-441c-98fe-15cb670f8ca3",
        "name": "approved assessment 1"
      },
      {
        "id": "7d9af425-7a70-4ecd-b88a-4dc27df233ef",
        "name": "approved assessment 2"
      }
    ],
    "startDate": "2021-03-03T14:40:49.0930127Z",
    "endDate": "2021-03-03T14:42:54.0930231Z",
    "elapsedSeconds": 125,
    "totalRecordCounts": [
      {
        "key": "analysisItems",
```

```
    "value": 12
  }
],
"totalPayloadCount": 1,
"resultsReady": true
},
"resultsSpecification": {
  "resultsScope": {
    "modelName": "my risk model",
    "resultsType": "RiskAssessmentApproved",
    "resultsTypeOptions": [],
    "returnFormat": "json"
  },
  "resultsFormulaGroups": [],
  "resultsKeyDataProperties": []
},
"resultsPayloadAddresses": [
  {
    "url": "/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0-5b2442eab5e5/payload/0"
  }
]
}
```

If you want to test your polling loop, you can force this call to return `resultsReady=false`. That way you could loop (for example, five times) with receiving false; then on the sixth call, make it respond as if results are ready, so you can break out of your loop and call for payloads as described in Step 5. To do this, put the "(TESTPOLL)" keyword into the ResultId guid parm:

GET/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0-(TESTPOLL)

It will return `resultsReady=False` to simulate the service still working on gathering results

```
{
  "resultsInfo": {
    "resultId": "033ed29f-5024-48d7-9cd0-(TESTPOLL)",
    "resultType": "RiskAssessmentApproved",
    "modelName": "my risk model",
    "assessments": [
      {
        "id": "7348cee5-efee-41ae-afa5-d1aa077d936e",
        "name": "approved assessment 1"
      },
      {
        "id": "77360ebe-44ca-4395-b6b6-58719fdcf8de",
        "name": "approved assessment 2"
      }
    ],
    "startDate": "2021-03-03T14:44:44.9556524Z",
    "endDate": "0001-01-01T00:00:00",
    "elapsedSeconds": 0,
    "totalRecordCounts": [],
    "totalPayloadCount": 0,
    "resultsReady": false
  },
  "resultsSpecification": {
```

```
"resultsScope": {  
  "modelName": "my risk model",  
  "resultsType": "RiskAssessmentApproved",  
  "resultsTypeOptions": [],  
  "returnFormat": "json"  
},  
"resultsFormulaGroups": [],  
"resultsKeyDataProperties": []  
},  
"resultsPayloadAddresses": []  
}
```

5. Requesting a payload(s)

Simulated payloads can be requested anytime for testing. Even without requesting the result job be started in prior calls (as described in Step 3). Both types can be requested and the number of records returned in a given payload can be simulated for testing. By default RiskAssessmentApproved results will be returned. Also by default the number of records in total in one payload are controlled by configuration parameters in the appsettings configuration. The following example will simulate one payload with 12 records total and all in payload #1.

```
"Startup": {  
  "IsMock": "True",  
  "MockPayloadRecordCount": 12,  
  "MockTotalRecordCount": 12,
```

- Default – Returns Risk Assessment approved results and the number contained in the settings as shown in the previous example.

GET/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0/payload/0

Will return 12 records of ResultType= RiskAssessmentApproved

- If you want to control the ResultType, you can simulate each resultType payload by putting keyword "(Analytical)" or "(Risk)" into the ResultId Guid for the call:

GET/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0
(analytical)/payload/0

Will return 12 records of ResultType=*AnalyticalAssessmentApproved*

- If you want to control the number of records of either type you receive in your payloads, you can simulate this by putting the keyword "{number}" (for example, (100)) into your ResultId Guid for the call:

GET/api/AssessmentResultsService/results/033ed29f-5024-48d7-9cd0(100)
(analytical)/payload/0

Will return 100 records in this payload of ResultType=
AnalyticalAssessmentApproved

For more information

For further information about the Synergi Pipeline Assessment Results Service API or to request assistance, contact your Solutions Support Manager or DNV - Digital Solutions Technical Support at software.support@dnv.com.