# *ServiceNow: Symphony Table Mapping V1.0*

***July 14, 2020***

## Notices



Following are policies pertaining to proprietary rights and trademarks.

### Proprietary Rights

The information contained in this document is proprietary and confidential to AVI-SPL, Inc., one or more of its affiliated entities (collectively “AVI-SPL”), or both.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from AVI-SPL.

All copyright, confidential information, patents, design rights, and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of AVI-SPL.

### Trademarks

The AVI-SPL name and the AVI-SPL logo are trademarks or registered trademarks of AVI-SPL, Inc.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

### Disclaimer

The information furnished herein is believed to be accurate and reliable. However, no responsibility is assumed by AVI-SPL for its use or for any infringements of patents or other rights of third parties resulting from its use.

SYMPHONY

# TABLE OF CONTENTS



  

# 1 INTRODUCTION

This document describes the Symphony table mappings utilized for ServiceNow. The document is intended for AVI-SPL customers and AVI-SPL personnel.

### 1.1.1 Ticket Status Mappings for ServiceNow

When status changes in **Symphony**, the mapping is as flows:

**Table 1 Symphony Status to ServiceNow State mapping**

| # | Symphony | ServiceNow |
|---|---|---|
| **1** | Open | Open |
| **2** | Close Pending | No change in ServiceNow status |
| **3** | Closed | Closed |

When the **state** changes in **ServiceNow**:

**Table 2 ServiceNow State to Symphony Status mapping**

| # | ServiceNow | Symphony |
|---|---|---|
| **1** | Open | Open |
| **2** | In progress | |
| **3** | On hold | |
| **4** | Resolved | Closed |
| **5** | Closed | |
| **6** | Canceled | |

**Conflict resolution rules:**

1. When ticket is closed in ServiceNow, it shall not be reopened based on Symphony monitoring event, (so, this situation should be exception to time-based conflict resolution).

  

### 1.1.2 Ticket Priority

Ticket priority mappings between Symphony and ServiceNow shall be as follows below. However, in ServiceNow by default, priority is read-only property which is calculated from provided **urgency** and **impact** (see information by following this link).

**Note**: Symphony, does NOT have either urgency or impact, however it is possible to turn this feature off in ServiceNow and directly provide priority in REST requests.

**Table 3 Ticket Priority in ServiceNow relative to Symphony Priority**

| # | ServiceNow Urgency | ServiceNow Impact | ServiceNow Priority | Symphony Priority |
|---|---|---|---|---|
| 1 | High | High | Critical | Critical |
| 2 | High | Medium | High | Major |
| 3 | High | Low | Moderate | Minor |
| 4 | Medium | High | High | Major |
| 5 | Medium | Medium | Moderate | Minor |
| 6 | Medium | Low | Low | Informational |
| 7 | Low | High | Moderate | Minor |
| 8 | Low | Medium | Low | Informational |
| 9 | Low | Low | Planning | Informational |

### 1.1.3 User Mapping Example

**Table 4 User Mapping**

| # | Symphony Email | 3rd party ID | 3rd party ID type | Source type | Notes |
|---|---|---|---|---|---|
| 1 | halynamudryk@avispl.com | halyna.mudryk | username | Symphony | 3rd party ID = halyna.mudryk will be mapped to halynamudryk@avispl.com and propagate to Symphony |
| 2 | halynamudryk@avispl.com | halyna.mudryk | username | ServiceNow | Symphony email = halyna.mudryk@avispl.com will be mapped to 3rd party ID = halyna.mudryk and propagate to ServiceNow with type = username.<br>User in ServiceNow will be found by criteria: username = halyna.mudryk |

  

| # | Symphony Email | 3rd party ID | 3rd party ID type | Source type | Notes |
|---|---|---|---|---|---|
| 3 | halynamudryk@avispl.com | halynamudryk@servicenow.com | email | ServiceNow | Symphony email = halyna.mudryk@avispl.com will be mapped to 3rd party ID = halynamudryk@servicenow.com and propagate to ServiceNow with type = email.<br>User in ServiceNow will be found by criteria: email = halynamudryk@servicenow.com |

### 1.1.4 Subject/Description Mappings for ServiceNow

Symphony has **subject** and **description**, while ServiceNow has just "**short description**" and set of **comments**.

To resolve this scenario:

- Put Symphony's **subject** to ServiceNow's "**short description"**
- Put Symphony's **description** to a ServiceNow's "**description**"

### 1.1.5 ServiceNow History

If you would like to review the changes made to the ServiceNow record after an update, the History can be viewed in Calendar or List form following the steps below:

1. Click the **Additional actions** menu icon.

**Figure 1 Additional actions**


2. Scroll to **History > List** for a detailed list of updates to a record.

  

**Figure 2 The List or Calendar shows the same info in different formats**

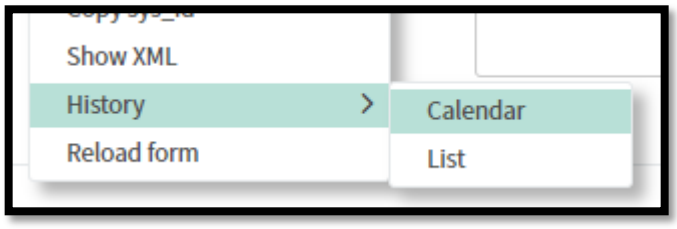


**Figure 3 Example of a record history**

Record History Details Page 1

**Report Title:** Record History Details
**Run Date and Time:** 2020-07-13 19:15:28 Eastern Daylight Time
**Run by:** System Administrator
**Table name:** sys_history_set

**Record History**

ID:
Incident: INC0010007
Table:
incident
Load time:
0 Seconds

**Related List Title:** History List
**Table name:** sys_history_line
**Query Condition:** Set = Incident: INC0010007
**Sort Order:** in descending order

34 History

| Field | Label | Old | New | ▼ Update time | User name |
|---|---|---|---|---|---|
| comments | Additional comments | | Device Information:<br>Name: Avaya Scopia Desktop<br>Type: Single Codecs<br>IP: 8.8.8.8<br>Serial Number: N/A<br>Manufacturer: Avaya<br>Model: Scopia Desktop | 2020-07-13 19:11:21 | symphony_admin<br>symphony_admin |
| activity_due | Activity due | 2020-07-13 21:11:19 | 2020-07-13 21:11:21 | 2020-07-13 19:11:21 | symphony_admin<br>symphony_admin |

  

## 1.2 SYMPHONY REST API DESIGN

**Note**: The following is an FYI regarding the Symphony REST API design and contains no action items.

**Authentication**

Current basic authentication approach adopted by Symphony.

**Response format**

All responses shall be in JSON format.

***POST requests (object creation)***

| # | Field | Data type | Mandatory | Notes |
|---|---|---|---|---|
| 1 | id | string | no | ID of newly created object or null if operation failed. |
| 2 | ts | long | no | Timestamp of creation instant.<br>If operation succeeds should be equal to last modified date of the relevant entity. |
| 2 | status | int | yes | 0 - operation succeeded<br>1 - operation failed |
| 3 | error | String | no | Error description, if available |

***PATCH requests (object updates)***

| # | Field | Data type | Mandatory | Notes |
|---|---|---|---|---|
| 1 | ts | long | no | Timestamp of update instant.<br>If operation succeeds should be equal to last modified date of the relevant entity. |
| 2 | status | int | yes | 0 - operation succeeded<br>1 - operation failed |
| 3 | error | String | no | Error description, if available |

***DELETE requests (removing objects)***

| # | Field | Data type | Mandatory | Notes |
|---|---|---|---|---|
| 1 | status | int | yes | 0 - operation succeeded<br>1 - operation failed |
| 2 | error | String | no | Error description, if available |

  

***GET requests (retrieving objects)***

| # | Field | Data type | Mandatory | Notes |
|---|---|---|---|---|
| **1** | status | int | yes | 0 - operation succeeded<br>1 - operation failed |
| **2** | error | String | no | Error description, if available |
| **3** | payload | object[] | yes | Empty list if request failed or list of objects applicable. |

**Ticket creation endpoint**

| Scope | Create ticket with comments (does NOT apply to attachments) |
|---|---|
| **HTTP method** | POST |
| **Payload format** | JSON |
| **URL format example** | /tickets |
| **Response payload** | Refer to POST response format specification above. |
| **Request payload** | Ticket object as per described in section 4.7.1 without optional fields |

**Ticket update endpoint**

| Scope | Update ticket, comments (does not apply to attachments) |
|---|---|
| **HTTP method** | PATCH |
| **Payload format** | JSON |
| **URL format example** | /tickets/{ticketID}<br>Where:<br>• {ticketID} is a ticket ID that gets updated |
| **Request payload** | Ticket object as per described in section 4.7.1 without optional fields |
| **Response payload** | Refer to PATCH response format specification above. |

**Add attachment**

| Scope | Add new attachment |
|---|---|
| **HTTP method** | POST |
| **Payload format** | multipart/form-data |
| **URL format example** | /tickets/{ticketID}/attachment<br>Where:<br>• {ticketID} is a ticket ID that gets updated |
| **Request payload** | Attachment object as per described in section 4.7.1 including optional fields |
| **Response payload** | Refer to POST response format specification above. |

  

### Update attachment

| Scope | Update attachment metadata (payload change is not possible) |
|---|---|
| **HTTP method** | PATCH |
| **Payload format** | multipart/form-data |
| **URL format example** | /tickets/{ticketID}/attachment/{attachmentID}<br>Where:<br>• {ticketID} is a ticket ID that gets updated<br>• {attachmentID} is an attachment ID that gets updated |
| **Request payload** | Attachment object as per described in section 4.7.1 including optional fields |
| **Response payload** | Refer to PATCH response format specification above. |

### Remove attachment

| Scope | Remove existing attachment |
|---|---|
| **HTTP method** | DELETE |
| **Payload format** | N/A |
| **URL format example** | /tickets/{ticketID}/attachment/{attachmentID}<br>Where:<br>• {ticketID} is a ticket ID that gets updated<br>• {attachmentID} is an attachment ID that gets updated |
| **Request payload** | N/A |
| **Response payload** | Refer to DELETE response format specification above. |

### Get ticket by ID

| Scope | Get specific ticket with comments and attachments metadata (without optional fields) |
|---|---|
| **HTTP method** | GET |
| **URL format example** | /tickets/{ticketID}<br>Where:<br>• {ticketID} is a ticket ID that gets updated |
| **Request payload** | N/A |
| **Response payload** | Refer to GET response format specification above.<br>payload field to contain one Ticket object in a list as per described in section 4.7.1 without optional fields |

  

**Get all tickets**

| Scope | Get all tickets with comments and attachments metadata (without optional fields) |
|---|---|
| **HTTP method** | GET |
| **Payload format** | N/A |
| **URL format example** | /tickets |
| **Payload structure** | Refer to GET response format specification above.<br>payload field to contain list of Ticket object as per described in section 4.7.1 without optional fields |

**Get changed tickets by timestamp**

| Scope | Get ticket with comments and attachments metadata (without optional fields) |
|---|---|
| **HTTP method** | GET |
| **Payload format** | N/A |
| **URL format example** | /tickets/?ts={timestamp}<br>Where:<br>• {timestamp} is a timestamp (long). If timestamp > last modified date of ticket, then ticket should be discarded and not returned |
| **Payload structure** | Refer to GET response format specification above.<br>payload field to contain Ticket objects as per described in section 4.7.1 without optional fields |

### 1.2.1 Symphony Callback API design

Symphony shall have possibility to invoke callback endpoint once ticket gets created, updated, or removed.

**Authentication**

Current basic authentication approach adopted by Symphony.

**Ticket changes**

| HTTP method | POST |
|---|---|
| **Payload format** | JSON |
| **Response payload** | N/A<br>Successful processing of change should be signified by HTTP 200 response code |
| **Request payload** | See below |

  

### Request payload specification

| # | Field | Data type | Mandatory | Notes |
|---|---|---|---|---|
| **1** | operation | int | yes | 0 - ticket created<br>1 - ticket updated<br>2 - ticket removed |
| **2** | ticketId | String | yes | |
| **3** | payload | object | no | Ticket object as per described in section 4.7.1 without optional fields.<br>Applicable for creation and update operations. |

### Resiliency

In case remote endpoint returns non 200 response code or interaction fails, Symphony shall implement retries mechanism with exponential back-off.

  