Post Resource Status API

Overview

Juvare provides the Post Resource Status application programming interface (API) as a web service that its clients and partners can use to enhance interoperability by updating resource status data in EMResource. Data is communicated using XML.

Benefits

The Post Resource Status API offers numerous benefits, including:

- **Security** – Data is communicated securely using the TLS 1.2 protocol, in compliance with XML data standards, and it is represented in an XML schema developed by Juvare.
- **Flexibility** – The API is designed to update virtually any status data for any set of resources in EMResource.
- **Configurability** – The API is configured based on the requirements and information you specify.

Terms

In relation to the Post Resource Status API document and the EMResource solution, there are some common terms that are helpful to keep in mind.

- **Resource** – An entity that reports a status. Resources can be facilities, organizations, and agencies that report status information on capabilities, services, and supplies.
- **Status Type** – A type of status that resources record, such as availability, date, quantity, score, or description.
- **Status** – A value or option that resources report for a status type.

Documentation

Juvare recommends using this document in conjunction with the details and procedures in the *Partner Interoperability Development Guide*. The interoperability guide describes web service interfaces and provides an example procedure for connecting to Juvare web services using a .NET web service client.

Gateway Partner Endpoints

Lab/QA: https://emresource.lab.juvare.com/partnergateway/PartnerEndpoint
Production: https://emresource.juvare.com/partnergateway/PartnerEndpoint

Requirements

The Post Resource Status API must be configured for EMResource by Juvare. The coinciding software vendor must have a username, password, action name, division name, resource IDs, status IDs, and status values.

Update Status Type

Request Header

```
<requestDefinition actionName='{Action Name}' systemName='EMResource'
                   divisionName='{Division Name}'>
</requestDefinition>
```

Request Body

```
<resources>
  <resource id='{Resource UUID}' updateType='U'>
    <statuses>
      <status>
            <statusName>{Status Key}</statusName>
            <statusValue>{Status Value}</statusValue>
            <statusValueType>{NUMRC|TEXT}</statusValueType>
      </status>
      <status>
            <statusName>{Status Key}</statusName>
            <statusValue>{Standard Status Name}</statusValue>
            <statusValueType>STATS</statusValueType>
      </status>
    </statuses>
  </resource>
</resources>
```

Response

```
<response>
   <exposition>
      <resource id="{Facility UUID}" updateType="U">
         <details>
            <errors>
                  <error id="externalID">{error message}</error>
            </errors>
            <totalSent>0</totalSent>
            <totalUpdated>0</totalUpdated>
         </details>
         <statuses>
            <errors/>
            <totalSent>0</totalSent>
```

```
            <totalUpdated>0</totalUpdated>
         </statuses>
      </resource>
   </exposition>
   <summary>
      <resourceFailures>1</resourceFailures>
      <resourceSuccesses>0</resourceSuccesses>
      <statusFailures>0</statusFailures>
      <statusSuccesses>0</statusSuccesses>
   </summary>
</response>
```

Values

Attribute	Format	Comment
actionName	Text	Uniquely identifies the web service and its scope
divisionName	Text	
username	Text	Username for authentication
password	Text	Password for authentication
resource id	UUID	Unique identifier of each resource (for example, facility)
statusName	Text	Unique identifier of type of status (for example, licensed beds)
statusValueType	Enumeration	Data type of the status type (NUMRC, TEXT, or STATS) DEPRECATED: statusValueType will be optional starting in EMResource v3.46
statusValue	Text|Number	Value of the status (for example, “555-555-1234” for ED Phone Number, 13 for licensed beds)

Implementation

Step 1: Configuration

Working in conjunction with Juvare, define the scope of resources and status types to include in the interface. Juvare will configure the interface and provide you with the specifications needed to make web service requests.

Step 2: Web Service Code and Connection

Write the code for your side of the interface, referring to example code and procedures available in the *Partner Interoperability Development Guide*. Make sure the code can connect to and invoke the Get Status API using the test credentials provided in the *Partner Interoperability Development Guide*.

Invoking the API requires a single Simple Object Access Protocol (SOAP) web service call. The SOAP header contains static parameters. The SOAP body contains the resources and

status types to be updated, along with the values to which you want to update them. You develop the code needed to generate the XML request and process the XML response that the web service returns to you.

With appropriate web service permissions, EMResource updates the status for all values that changed. If you do not have permission to update a specific status or the value has not changed, EMResource does not update the status.

EMResource processes all authorized and valid updates and skips the ones that are not. The response includes a report of which ones succeeded and an error message for each failure.

Technical Support

Juvare supports your development efforts by providing the *Partner Interoperability Development Guide*, technical documentation on the specific API, the XML schema document, and examples of request statements. We are also able to assist you with testing the interface.

Request and Response Examples

You need to submit a request to update data in EMResource.

Request Header Example

```
<requestDefinition actionName='postResourceStatus'
                   systemName='EMResource' divisionName='DivisionABC'>
</requestDefinition>
```

The configured web service validates the request, retrieves the data, and delivers it in XML.

Request Body Example

You need to develop and use a processing code to parse and process the XML.

```
<resources>
  <resource id='976eb6c3-xxcd-4b55-ae94-4a25bf8d17d6' updateType='U'>
    <statuses>
      <status>
            <statusName>Avail-AdultICU</statusName>
            <statusValue>111</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-NICU</statusName>
            <statusValue>222</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
```

```
            <statusName>Avail-Pediatric</statusName>
            <statusValue>333</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-PICU</statusName>
            <statusValue>444</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-VentsTotal</statusName>
            <statusValue>555</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
    </statuses>
  </resource>
  <resource id='abceb6c3-xxcd-4b55-ae94-4a25bf8d17d6' updateType='U'>
    <statuses>
      <status>
            <statusName>Avail-AdultICU</statusName>
            <statusValue>11</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-NICU</statusName>
            <statusValue>22</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-Pediatric</statusName>
            <statusValue>33</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-PICU</statusName>
            <statusValue>44</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
      <status>
            <statusName>Avail-VentsTotal</statusName>
            <statusValue>55</statusValue>
            <statusValueType>NUMRC</statusValueType>
      </status>
    </statuses>
  </resource>
```

Response Body Example

The HTTP response contains a summary of how much of your request succeeded, and an error message for each failure. You need to develop code to parse and process the XML.

```
<response>
   <exposition>
         <resource id="4f64566aa3744fc8993c0a7a049bb3e5"
                  name="My Hospital" region="Region1" updateType="U">
               <details>
                     <errors/>
                     <totalSent>0</totalSent>
                     <totalUpdated>0</totalUpdated>
               </details>
               <statuses>
                     <errors>
                           <error id="Invalid-Key">Invalid-Key=-Unknown-; User
                           may not update status</error>
                     </errors>
                     <totalSent>3</totalSent>
                     <totalUpdated>1</totalUpdated>
               </statuses>
         </resource>
   </exposition>
   <summary>
         <resourceFailures>0</resourceFailures>
         <resourceSuccesses>1</resourceSuccesses>
         <statusFailures>2</statusFailures>
         <statusSuccesses>1</statusSuccesses>
   </summary>
</response>
```

Setup

Values used in the API setup are unique to you, the client, and must be obtained from Juvare. These values include:

- Web service parameters (actionName and divisionName)
- Web service user credentials (username and password)
- Test user credentials (username and password)
- Resource IDs (for example, hospital A, B, and C, and clinic 1)
- Status Types (also known as data elements)

Verification

After running the API, log in to EMResource using the test user credentials to verify the status changes.

Need Help?

For more information, contact the Juvare Support Center by sending an email to support@juvare.com or by calling 877-771-0911.