PI Web API Reference

Contents

PI Web API Reference

PI Web API Reference

The PI Web API is a RESTful interface to the PI system. It gives client applications read and write access to their AF and PI data over HTTPS. Use the links on the left to learn about the PI Web API in more detail:

The **Getting Started** section introduces the concepts of a RESTful service in the context of the PI Web API and describes some important general constructs and principles found throughout the API. It concludes with a tutorial demonstrating a simple HTML/CSS/JavaScript client that uses the PI Web API.

The **Controllers** section lists the top-level endpoints provided by the service. Each controller's detail page provides links to the methods exposed by the controller. Use these pages as a reference when programming client applications.

The **Topics** section provides links to detailed specifications of options and features that appear throughout the PI Web API. For example, several methods use time strings for input, output, and query filtering. The Time Strings topic describes the use and formatting of these strings.

The **Changelog** describes the incremental changes included in each release of the PI Web API.

Getting Started with the PI Web API

The PI Web API is a RESTful interface to the PI system. It gives client applications read and write access to their AF and PI data over HTTPS. This getting started guide starts with an overview of the constructs and principles that you'll encounter as you use the API, moves on to examples in the form of sample requests and a simple demo application, and concludes with a discussion of some more specific topics that may enrich your understanding of the API.

REST Principles

REST stands for 'representational state transfer.' In the context of the PI Web API, this means that the API is:

- Stateless

  The PI Web API is stateless. This means that the service retains no observable knowledge of clients across requests. Each request is an independent transaction between the client and the server. One important consequence of this property is that the check out - make changes - check in transaction pattern found in many database-related APIs (e.g. the AFSDK) is not exposed by the PI Web API. Rather, each request encapsulates this pattern internally.

- Resource-oriented

  Interaction with the PI Web API is organized around resources. A resource is a structured piece of data that represents an object in one of the PI Systems connected to the Web API. Most important PI and AF objects, such as Asset Servers, Data Servers, points, elements, attributes, event frames, and so on, map to resources in the PI Web API. Four primary operations are used to interact with these resources: create, read, update, and delete (often abbreviated as CRUD).

- Navigable by links

  Links (also called hypermedia) capture the organization of the resources exposed by the Web API. You're probably familiar with the hierarchical structure of AF objects: Asset Servers contain databases, which contain elements, which contain attributes, and so on. Links express these relationships. For example, an Asset Server resource links to the collection of databases it contains, and each database links to its parent Asset Server.

WebID

Resources that support CRUD operations are the primary objects in the PI Web API. To allow clients to address these resources, the PI Web API must identify them in a way that is both persistent (so that clients can address known resources consistently over time) and URL-safe (since resources are specified by URLs). Every primary resource in the PI Web API is associated with a WebID, which is an identifier that meets these criteria. Client applications should treat WebIDs as opaque identifiers. Because they are persistent, clients may cache URLs containing WebIDs.

HATEOAS

As discussed above, the resources exposed by the PI Web API are connected by links. This means that client applications should rarely need to construct resource URLs. Rather, they should use Hypermedia as the Engine of Application State (HATEOAS). There are several entry points to the application, including the root of the API and the various GetByPath methods. Once 'inside' the system, the client application should access related resources via the links.

HTTP Verbs

Every resource exposed by the PI Web API supports some subset of the CRUD operations discussed above. Create corresponds to POST, read to GET, update to PUT (when the method requires the complete resource definition) or PATCH (when the method accepts a partial resource definition), and delete to DELETE.

URL Parameters

Several PI Web API GET methods accept URL query parameters. In general, these specify options or serve as filters on the response. For example, element search (GET assetdatabase/{webId}/elements) takes various parameters that specify which elements to return and how to order the returned elements.

JSON

JavaScript Object Notation (JSON) is the primary media type supported by the PI Web API. You can get a feel for JSON by examining the samples provided in the documentation and exploring the API in your browser. Most common client application languages and frameworks provide libraries to convert between language/framework primitives and JSON strings.

Status Codes

HTTP status codes provide information about the success or failure of a request. The PI Web API adheres to the standard semantics of these codes as closely as possible. A coarse distinction is that 200-level status codes indicate success, 400-level status codes indicate user error, and 500-level status codes indicate server error. 400- and 500-level codes are generally accompanied by a response body providing a friendly error message to assist with debugging. One special case worth noting is that the 201 status code is returned in response to a POST, when a resource has been created. In this case, the response will contain a Location header with the WebID-based URL, which the client may use to access the newly-created resource.

The Structure of a PI Web API Request

In this section, we'll look at a GET request to the PI Web API. Client applications will generally use libraries to take care of the details of constructing HTTP requests, but an understanding of the information contained in a request, coupled with a request inspector like Fiddler, is useful for debugging. The following is a complete GET request to the API:

```
GET https://myserver/piwebapi/assetdatabases/D0NxzXSxtlKkGzAhZfHOB-KAQLhZ5
wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg HTTP/1.1
Host: myserver
Accept: application/json
```

The first line specifies the GET method, the resource URL, and the protocol version. The next two lines are request headers. The second line specifies the host addressed by the client, and the third line specifies that the client expects to receive a response formatted as JSON. The response looks like this:

```
HTTP/1.1 200 OK
Content-Length: 1783
Content-Type: application/json; charset=utf-8
Server: Microsoft-HTTPAPI/2.0
Date: Wed, 01 Oct 2014 14:24:13 GMT

{
  "WebId": "D0NxzXSxtlKkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVH
UkVFTg",
  "Id": "e759b840-d40a-4cf9-910d-09f307fcc654",
  "Name": "NuGreen",
  "Description": "PI BI Project Asset Model",
  "Path": "\\\\MyAFServer\\NuGreen",
  "Links": {
    "Self": "https://myserver/piwebapi/assetdatabases/D0NxzXSxtlKkGzAhZfHO
B-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg",
    "Elements":"https://myserver/piwebapi/assetdatabases/D0NxzXSxtlKkGzAhZ
fHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/elements",
    "ElementTemplates":"https://myserver/piwebapi/assetdatabases/D0NxzXSxt
lKkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/elementtemplat
es",
    "EventFrames":"https://myserver/piwebapi/assetdatabases/D0NxzXSxtlKkGz
AhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/eventframes",
    "AssetServer":"https://myserver/piwebapi/assetservers/S0NxzXSxtlKkGzAh
ZfHOB-KAUEhMQUZTMDQ",
    "ElementCategories":"https://myserver/piwebapi/assetdatabases/D0NxzXSx
tlKkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/elementcatego
ries",
    "AttributeCategories":"https://myserver/piwebapi/assetdatabases/D0NxzX
SxtlKkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/attributeca
tegories",
    "TableCategories":"https://myserver/piwebapi/assetdatabases/D0NxzXSxtl
KkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/tablecategories
",
    "EnumerationSets":"https://myserver/piwebapi/assetdatabases/D0NxzXSxtl
KkGzAhZfHOB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/enumerationsets
",
    "Tables":"https://myserver/piwebapi/assetdatabases/D0NxzXSxtlKkGzAhZfH
OB-KAQLhZ5wrU-UyRDQnzB_zGVAUEhMQUZTMDRcTlVHUkVFTg/tables"
```

```
  }
}
```

The first line specifies the protocol version and the response status code and reason phrase, indicating the overall success or failure of the request. The following four lines are response headers. Lines 2-3 provide information about the content of the response. Lines 4-5 contain information about the server and the time of the request. Below the headers is the response body, a JSON string providing information about the database. Notice the WebID and the links specifying the URLs of related resources and resource collections.

A Simple Application

In this section, we'll develop a JavaScript application for exploring the AF hierarchy of your PI System. A basic knowledge of JavaScript, HTML, and CSS is assumed. The following is the complete code. If you're viewing this on a live instance of PI Web API, click here to view a live, lightly-modified version of this application.

```
<!DOCTYPE html>
<html>
<head>
  <title>AF Hierarchy Viewer↙title>
  <script src="jquery.min.js">↙script>
  <script type="text/javascript">
    var childrenMap = {
      PISystems: ['AssetServers'],
      AssetServers: ['Databases'],
      Databases: ['Elements'],
      Elements: ['Elements', 'Attributes'],
      Attributes: ['Attributes']
    };

    function node(name, type, links, parentDiv) {
      let nameSpan = $('<span ↗').addClass(type).text(name);
      let flipper = $('<span ↗').addClass('flipper').text('+').click(flip.bin
d(this, this));
      let indent = $('<div ↗').addClass('indented').hide();

      this.type = type;
      this.links = links;
      this.flipper = flipper;
      parentDiv.append(this.flipper).append(nameSpan).append('<br↗');
      this.div = indent.appendTo(parentDiv);
      resizeHeader();
    }

    function loadChildren(n) {
      n.loaded = true;
      childrenMap[n.type].forEach(function(childCollection) {
        $.get(n.links[childCollection], function(collection) {
          n[childCollection] = collection.Items.map(function (item) {
```

```
            return new node(item.Name, childCollection, item.Links, n.div)
;
          });
        });
      });
    }

    function flip(n) {
      if (!n.loaded) { loadChildren(n); }
      n.flipper.html(n.flipper.html() == '+' ? '-' : '+');
      n.div.toggle();
    }

    $(function() {
      root = new node('PI Systems', 'PISystems',
        { AssetServers: 'https://myserver/piwebapi/assetservers' }, $("#ro
ot"));
    });
  ↙script>
  <style type="text/css">
    div {
      left: 10px;
      position: relative;
    }
    .flipper {
      cursor: pointer;
    }
  ↙style>
↙head>
<body>
  <div id="root">↙div>
↙body>
↙html>
```

The first few lines contain HTML boilerplate to set up the page and load jQuery. The section containing the JavaScript is the meat of the application.

`childrenMap` is a map from collection type to children collection types. For example, an 'AssetServers' collection contains objects (Asset Servers) whose child collections are 'Databases'. Note that the values in this map are chosen to correspond to the keys in the links collections returned by the Web API.

`node` creates a new node in the displayed AF hierarchy. It stores the collection type and links associated with the object used to create the node, then appends a new expandable element to the DOM.

`loadChildren` performs the AJAX GET requests to load the given node's children into the application. JavaScript automatically maps the JSON responses from the WebID to native JavaScript objects that we can access as in `collection.Items` and `item.Name`. A new node is created for each child object. As mentioned above, the values in `childrenMap` correspond to keys in the links collections. This property is used when determining the resource for the GET request: `n.links[childCollection]`.

`flip` is the callback associated with clicking the '+'/'-' symbol next to each expandable node. If the node's children haven't been loaded yet, they are. Then, the '+'/'-' symbol and visibility of the child div are toggled.

The last bit of the script, `$(function() {...`, executes as soon as the page is loaded. It establishes the root node of the AF hierarchy. The page's source ends with some basic CSS styling and the HTML to set up the root node.

In this example, we can see several of the PI Web API principles listed above in action. Notice that the only hard-coded URL is the Asset Servers root. All other resources are retrieved by following links. You can use your browser console to examine the AJAX requests made by the application. You'll see the structure of the JSON objects returned by the Web API, including the links collections. You'll also notice the WebIDs embedded in the URLs. A richer application might include create, update, or delete functionality, the ability to do filtered reads using query parameters, and more robust error handling (the sample application fails silently for non-200-class responses).

Other Concepts

Caching and Concurrency

The PI Web API caches AF data to improve performance. It maintains up to two caches for each recently-connected user: one for reads and one for writes. The read cache can service multiple simultaneous requests, but each write request requires exclusive access to the write cache. Some notable consequences and caveats around this design are:

- If you're using anonymous authentication, every read/write request is served from the same read/write cache.
- The `Cache-Control: no-cache` request header can be used to force the AF cache to refresh from the backend Asset Server. This operation requires an exclusive lock on the cache, however, so it should be used sparingly.
- A request that writes time series data is not a writer request, because time series data reads and writes are always served directly from the backend Data Server.

Timezones

On output, the PI Web API always displays times as ISO 8601 UTC strings. On input, several options are available. You may specify any ISO 8601 string as UTC, or with or without an offset. You may also specify PI times, such as '*' or 'T'. Time strings specified as UTC or with an offset are stored unaltered. Times without an offset and PI times are resolved relative to the timezone of the backend server on which the data is stored (i.e., not relative to the time zone of the client or the PI Web API).

Streams and Values

Data retrieval from attributes with a data reference and PI points is unified under the /streams... and /streamsets... (for bulk retrieval) endpoints. Use stream methods to retrieve recorded, plot, interpolated, and summary values for time series data. Values of attributes that do not reference time series data can be retrieved from /attributes/{webId}/value.

Stream Sets

For most use cases, the standard single-resource CRUD operations exposed by the PI Web API should be sufficient. To accommodate advanced use cases that require reading from or writing to multiple streams in a single request, the PI Web API exposes stream set resources. A stream set is a set of streams related by a common ancestor element, event frame, or attribute. The stream set endpoints allow reads and writes of time-series data to be made against multiple streams in a single call. The PI Web API also exposes 'ad-hoc' stream set endpoints, which permit reading and writing of data against multiple independent streams (i.e. streams that don't necessarily share a common ancestor).

Additional Resources

Additional information related to using and administering the PI Web API is available from the following sources:

- The PI Developers Club community, which has subscription-based resources to help you with the programming and integration of OSIsoft products, including tutorials, forums, and sample code. Many samples and tutorials hosted on PI Developers Club assume the presence of the *NuGreen* database, which you can run yourself by importing the NuGreen XML into your PI Asset Server.
- The PI Web API Users Guide, hosted in , contains detailed information on how to run and administer the product. AVEVA Documentation also contains information on how to install, configure, maintain, and use other components in the PI System.

Changelog

Items listed in **bold** are considered potentially breaking changes from the previous released version. Potentially breaking changes include:

- Changes to the structure of returned objects
  - Addition/removal of properties or links
  - Changes to the structure of composed types (e.g. collections)
- Status code changes
- Changes to default parameter values

We make a special effort not to remove properties or links, change the structure of composed types, change the class of status codes (e.g. 200 class to 400 class), or change default parameter values.

Clients should be prepared to handle changes to status codes within the same class, especially the 400 and 500 classes, and to handle the addition of properties and links to response objects.

PI Web API 2023

New Features:

- Bearer authentication is reworked and improved.
  - PI Web API can use access tokens to authenticate the user's Identity. In this release, AVEVA Identity Manager (AIM) server is the only supported Identity Provider.
  - PI Web API also supports Open ID Connect (OIDC) Authorization Code flow.
- Allow PI Web API Administrators to remove a cache instance via the Remove Cache Instance.
- **PI Web API's OIDC well-known endpoint is now reworked to retrieve the discovery document from AIM (Aveva Identity Manager)**

Data model changes:

- **Change the property name from 'User' to 'UserName' on objects of cache instance**
- **Change the property name from 'Sid' to 'UserId' on objects of cache instance**
- **Expose 'IsWindowsIdentity' property on objects of cache instance**

PI Web API 2021 SP3

New Features:

- PI Point queries are exposed via the Get Points By Search endpoint.
  - The query syntax follows the AFSDK Point Query syntax.
  - This endpoint utilizes the underlying AFSDK PI Point search.

Bug Fixes:

- Fixed an issue with incorrect child event frames being returned with ambiguous search root.

PI Web API 2021 SP2

New Features:

- The Admin Utility GUI now allows the certificate validation check to be bypassed.
  - Previously, the Admin Utility GUI only allowed a certificate to be selected if it used the SHA-2 signing algorithm and was trusted by the local machine. To use an untrusted certificate, the system administrator had to use the Admin Utility's command-line interface.
  - Now, a system administrator can optionally choose to disable the check when using the Admin Utility GUI, and can select any certificate that has a private key.
  - Note that bypassing the certificate validation check allows a system administrator to select an insecure certificate. When bypassing the validation check, system administrators should only use certificates they know to be secure.
- Ad-hoc stream set requests that include duplicate Web IDs will now succeed.
  - Previously, an ad-hoc stream set request that included duplicate Web IDs could fail. Now, the operation will deduplicate when retrieving resources.
  - Note that deduplicating Web IDs may cause the number of results to not match the number of supplied Web IDs. When sending duplicate Web IDs, client applications should be careful to index into the results using the Web ID rather than an offset.

Bug Fixes:

- Fixed a race condition where a successful request could incorrectly report a failure due to a disposed user identity.

PI Web API 2021 SP1

New Features:

- The 'EnableBuiltInHelp' configuration setting has been added.
  - When set to `true`, the PI Web API's built-in help system will be enabled.
  - When set to `false`, the PI Web API's built-in help system will be disabled. Attempts to access the built-in help system will be redirected to the equivalent public AVEVA documentation webpage.
  - This setting is `false` by default.

Other Changes:

- **The `CreateSearchByAttribute` and `ExecuteSearchByAttribute` endpoints return more information about what caused the request to fail.**
  - Previously, when a request using these endpoints failed due to a malformed query parameter or request content, the returned error did not describe why the query parameter or request content was malformed. Now, an additional error message describes why the query parameter or request content was malformed.

- Because the human-readable error message has changed, this is considered a breaking change. If your application does not programmatically parse the returned error messages, this is not a breaking change.

PI Web API 2021

New Features:

- Added the Performance Metrics Controller
  - Performance Metrics allows system administrators to view PI Web API usage, system load, and aggregate request performance.
  - For more information, see the Performance Metrics topic.
- Allow Point Source filtering when listing a Data Server's PI Points.

Bug Fixes:

- Channels now emit data source communication errors to clients.
  - For more information see the Error Handling section in Channels topic
- The PI Web API built-in help system now loads correctly when OMF is the only installed plugin.
- Request IDs are now preserved across the PI Web API's Admin, Analytic, and Debug Event Logs.
- Invalid configuration options now cause error events to be logged to the Admin Event Log. Previously, informational events would be logged.
- Users' permission settings no longer affect access to the Known Servers Table.

Other Changes:

- `InstanceConfiguration` **endpoint has been renamed to** `ActiveConfiguration`
  - `ActiveConfiguration` now shows all configuration settings, even those that have not yet been configured. If a setting has not been configured, the default value is shown.
  - The `InstanceConfiguration` endpoint is still available, but will return the same results as the `ActiveConfiguration` endpoint.
- **Unrecognized AF and DA servers are no longer automatically added by default.**
  - Previously, if a request tried to access an AF or DA server that was not recognized by the PI Web API server, the PI Web API would connect to the server and automatically add it to the set of known servers.
  - To re-enable automatically adding unrecognized servers, enable the 'AllowUnrecognizedAssetServers' and 'AllowUnrecognizedDataArchives' settings.
  - For more information, consult the PI Web API manual.
- **The 'BearerEnableJwt' configuration setting has been removed. The authentication flow controlled by this setting has been removed.**
- **The default values of the 'BearerValidIssuer' and 'BearerValidAudiences' settings have changed.**
  - Previously, these settings had a default value of `null`. These settings now have default values that are intentionally invalid, to prevent accidental misconfiguration.
  - Existing PI Web API configurations may require changes after upgrading due to these new default values. For more information, consult the PI Web API manual.
- 'AuthenticationMethods' configuration values are now case-insensitive.

PI Web API 2019 SP1

Other Changes:

- Added Content Security Policy to defend against Cross Site Scripting
  - The default Content Security Policy can be overridden using the Custom Headers feature to specify a reporting URI. For details, consult the PI Web API manual.
- Security & Bug Fixes

PI Web API 2019

New features:

- Open Message Format (OMF) Services
  - OMF v1.1 data ingress through PI Web API endpoint now supported over HTTPS when Open Message Format Services are installed
    - Support for `create` and `delete` actions
    - `update` actions are not implemented
    - For more information, see the OMF Endpoint Notes topic
      - This link is only available if the OMF plugin has been installed by your system administrator
- Stream Updates has been made part of the Core Services feature, and has graduated from Community Technology Preview (CTP). For more information, see the Stream Updates topic

Other Changes:

- Security & Bug Fixes

PI Web API 2018 SP1

New features:

- Notifications
  - Read for child NotificationContactTemplates
  - Read for NotificationPlugIn
  - Read, create, update, and delete for SecurityEntries on NotificationRules, NotificationRuleTemplates and NotificationContactTemplates
  - Create, update and delete for NotificationRules, NotificationRulesTemplates, NotificationContactTemplates and NotificationRuleSubscribers
  - GetByPath endpoints for NotificationContactTemplates, NotificationRules, and NotificationRuleTemplates
  - Search endpoint for NotificationContactTemplates
- Stream Sets GetJoined Endpoint
  - Endpoint for StreamSets that returns a set of recorded values (x-axis) with another set of data for any number of streams (Y, Y', Y''... axis) that are interpolated based on the points returned for the x-axis
- Stream Updates (CTP)
  - Metadata changes from AF Servers are handled

- Selected fields are honored
- Responses now include PreviousEventAction information
- Error messages are shown for markers that are no longer valid

Data model changes:

- **Expose 'Paths' and 'DataReference' properties on objects of attribute**
- **Expose 'Paths' property on objects of element**

Other changes:

- The version of Web ID returned by PI Web API can be configured
  - PI Web API instances that run 2018 SP1 can now work together with older versions behind a load balancer since they can be configured to return the same version of Web ID
- Removed OpenAPI/Swagger specification endpoint
- Bug fixes

PI Web API 2018

New features:

- Read for notification contact templates, notification rules, notification rule subscribers and notification rule templates
- Read, create, update and delete for annotation attachments on event frames
- Retrieve relative paths of an element
- Retrieve full inheritance branch of an element template
- Allow reading annotations of a stream using the "associations" parameter
- Allow showing all child attribute templates of an element template
- Add parameter support for tables
- Filter attributes by trait and trait category
- Support health traits
- Incrementally receive updates from streams with Stream Updates (CTP)

Data model changes:

- **Expose 'ServerTime' property on objects of asset server, data server and system**
- **Expose 'DisplayDigits', 'Span' and 'Zero' properties on objects of attribute and point**
- **Expose 'DefaultUnitsNameAbbreviation' property on objects of attribute and attribute template**

Other changes:

- Add 'PreflightMaxAge' configuration setting

PI Web API 2017 R2 SP1

Bug fixes; no significant behavior changes.

PI Web API 2017 R2

New features:

- Web ID 2.0
- AFSearch based Element search
- AFSearch based Event Frame search
- AFSearch based Attribute search
- AFSearch based Analysis search
- AFSearch based Analysis Template search
- Retrieve the list of all available actions
- Add Sync Time parameters to all "Interpolated" endpoints
- Allow opting out of getting the search results for "CreateSearchByAttribute" endpoints
- Batch and "CreateSearchByAttribute" endpoints work in Read-only mode
- Channels support heartbeat of empty data frames

Data model changes:

- **Expose 'WebException' property on all PI Web API service responses. Any response containing a 'WebException' indicates that PI Web API encountered an unhandled error during the transfer of the response stream. These errors only occur after PI Web API has sent the client a successful HTTP status code. Responses will not contain a 'WebException' if no error occurred. The 'WebException' property will be present at the top level of all response objects, except during responses from the Stream and Stream Set controller responses, in which case each stream value may present with an 'Errors' field (see 'Errors' below).** For more information see the Error Handling topic.
- **Expose 'Errors' property on time series data responses from the Stream and Stream Set controller. Any time series value containing an 'Errors' property indicates PI Web API encountered an handled error during the transfer of the response stream. Values will not contain a 'Errors' property if no error occurred. Each object in the list of 'Errors' contains which field name caused an error during this time, as well as the exception message. Unlike 'WebException', which is a single property found at the top level of the response object, an 'Errors' property may be present on each stream value. If a stream value contains an 'Errors' property, then its value will be 'null'.** For more information see the Error Handling topic.

Other changes:

- **Previously, the Stream controller would return an HTTP error response if any of the time series data values to be returned contained an AFSDK exception for its value. This has been changed. A full time series data payload will now be returned, and any values that contain an AFSDK exception as a value will have a value of 'null'. The exception will now be included in the 'Errors' property.**
- **The "/value", "/recordedattime", and "/end" actions on the Stream Controller will now verify if the value to be returned contains an AFSDK or PI Web API exception. These endpoints now return an error HTTP Status Code if any error occurs.**

PI Web API 2017

New features:

- Support of claims-based authentication using ADFS, Azure AD, and PingFederate
- Search for event frames by 'Severity', 'IsAcknowledged', and 'CanBeAcknowledged' properties
- Retrieve recorded values at multiple timestamps for a stream or stream set
- Expose PI Web API 'Status' on System
- **Protection against Cross-Site Request Forgery attacks**
- Swagger specification
- Improved point attribute filtering
- New summary type: 'TotalWithUOM'

Data model changes:

- **Expose AFSecurity in EventFrames**
- **Expose 'TimeZone' and 'ConvertToLocalTime' properties of tables**
- **Expose Product Title of PI Web API versions**
- **Hide Point link on AFAttributes with invalid PI Point data references.**

Other changes:

- Add 'EnableCSRFDefense' and 'XFrameOptions' configuration settings
- Merge Channel controller to the core services

PI Web API 2016 R2

New features:

- Read, create, update and delete for security identities
- Read, create, update and delete for security mappings
- Read, create, update and delete for security entries
- Read, create, update and delete for annotations on event frames
- Manage attribute traits
- Allow to add an existing element as a child to another element
- Allow to provide time zone information as query parameter for stream and streamset resources
- Support Gzip compression for inbound data

Data model changes:

- **Expose 'IsAnnotated' and 'IsLocked' properties on objects of event frame**
- **Expose 'Descriptor', 'DigitalSet', 'EngUnits' and 'Step' properties on objects of point**
- **Expose 'TraitName' property on objects of attribute**
- **Expose 'TraitName' property on objects of attribute template**

PI Web API 2016 SP1

New features:

- Read, create, update and delete for analyses
- Read, create, update and delete for analysis categories

- Read, create, update and delete for analysis templates
- Read, create, update and delete for analysis rules
- Read, create, update and delete for time rules
- Read analysis rule plug-ins
- Read time rule plug-ins

PI Web API 2016

New features:

- Search element attributes matching various filters
- Search event frame attributes matching various filters
- Retrieve security rights of a user for an AF securable resource
- Retrieve multiple attributes by web id or path
- Retrieve multiple points by web id or path
- Retrieve (optionally) the attribute templates from the ancestors of the current element template
- White list selection of response fields
- A new type of batch sub-request: request template
- Add two web id aliases: "S00" for the local (PI Web API host) asset server and "S0D" for the default asset server

Data model changes:

- **Expose 'ExtendedProperties' property on objects of element template, element, event frame, asset database and asset server**
- **Expose 'IsManualDataEntry' property on objects of attribute**
- **Expose 'InstanceType' property on objects of element template**
- **Expose 'NamingPattern' property on objects of element template**
- **Expose 'DefaultValue' property on objects of attribute template**
- **Expose 'HasChildren' property on objects of attribute template, attribute, element and event frame**
- **Expose 'BaseTemplate' property on objects of element template**
- **Expose 'Acknowledgement' property on objects of event frame**
- **Expose 'Severity' property on objects of event frame**
- **Add 'Categories' link on objects of attribute template and element template**
- **Add 'Path' property for the stream in StreamSet responses**

Other changes:

- Add 'CorsExposedHeaders' configuration setting
- Merge Batch controller to the core services

PI Web API 2015 R3

New features:

- Search elements or event frames by attribute value
- CreateConfig for all attributes of elements or event frames

- Create, read, update, and delete for event frames' referenced elements
- Allow any HTTP method request as HTTP POST
- Get a single recorded value at a given time for Streams or StreamSets
- Get interpolated values at times for Streams or StreamSets
- Global request content length limit
- Batch (CTP)
- Channel (CTP)

Data model changes:

- **Exposed property 'RefElementWebIds' on event frame objects**
- Integrate AFJson library

Other behavior changes:

- None

PI Web API 2015 R2

New features:

- Bulk writes of time-series data (single and multiple value)
- Ad-hoc bulk reads and writes of time-series data
- Future PI points
- End-of-stream values for streams and stream sets
- Filtering of hidden and excluded attributes
- Event frame value capture

Data model changes:

- **Exposed properties 'IsExcluded' and 'IsHidden' on attribute and attribute template objects**
- **Exposed property 'Future' on point objects**
- **Exposed property 'IsConnected' on Asset and Data Server objects**
- **Exposed property 'AreValuesCaptured' on event frame objects**

Other behavior changes:

- None

PI Web API 2015

New features:

- Create, update, and delete for databases
- Create, update, and delete for event frames
- Create, read, update, and delete for enumeration sets and digital states
- Create, read, updated, and delete for element categories and attribute categories

- Create, read, update, and delete for tables and table categories
- Create, read, update, and delete for units and unit classes
- Create, update, and delete for points
- Read and update for point attributes
- Bulk retrieval of time-series data:
  - Current values
  - Recorded values
  - Interpolated values
  - Plot values
  - Summary values
- AF calculations
- Summary value filtering (streams and stream sets)
- Additional event frame search modes

Data model changes:

- **Exposed properties 'Type' and 'TypeQualifier' on attribute and attribute template objects**
- **Exposed property 'Step' on attribute objects**
- **Exposed link 'Point' on attributes with point data references**
- **Exposed link 'DefaultAttribute' on event frames with a default attribute**
- **Exposed link 'Categories' on event frames**

Other behavior changes:

- **The default stream value update option changed from 'NoReplace' to 'Replace'.**
- **To accommodate filtering, stream summary value retrieval may now return error values as part of a successful (200) response. An error value may take the form of a string or an object with a single property, 'Errors', whose value is an array of error strings.**

PI Web API 2014 R2 Update 1

Bug fixes; no significant behavior changes.

PI Web API 2014 R2

New features:

- Resources for indexed search

Data model changes:

- None

Other behavior changes:

- None

PI Web API 2014

- Create, read, update, and delete for elements
- Create, read, update, and delete for attributes
- Retrieval of time-series data:
  - Value at a timestamp, or current value
  - Recorded values
  - Interpolated values
  - Plot values
  - Summary values
- Single-value writes of time-series data
- Read for event frames
- Read for Asset Servers and databases
- Read for Data Servers and points

Changelog

Items listed in **bold** are considered potentially breaking changes from the previous released version. Potentially breaking changes include:

- Changes to the structure of returned objects
  - Addition/removal of properties or links
  - Changes to the structure of composed types (e.g. collections)
- Status code changes
- Changes to default parameter values

We make a special effort not to remove properties or links, change the structure of composed types, change the class of status codes (e.g. 200 class to 400 class), or change default parameter values.

Clients should be prepared to handle changes to status codes within the same class, especially the 400 and 500 classes, and to handle the addition of properties and links to response objects.

PI Web API OMF Services 2023

New features:

Bug Fixes:

PI Web API OMF Services 2021 SP3

For more information, see the PI Web API OMF Services 2021 SP3 detailed changes topic.

New features:

- Simplified Type delete
  - Only TypeID is required to delete a type - other fields will be ignored.
- Simplified Container delete

    - Only Container and TypeID are required to delete a container - other fields will be ignored.
- Simplified Static Data delete
    - Only TypeID and Values are required to delete static data - other fields will be ignored.
- Case Insensitivity
    - Container ID's are now case insensitive.
    - Type ID's are now case insensitive.
    - Static Data indicies are now case insensitive.

Enhancements:

- Container Create/Update will not overwrite Step point attribute of OMF PI Points when taking over existing PI Point.
- Contianer Update will not overwrite DataSource point attribute of OMF PI Points when "datasource" field is not specified in OMF messages.
- Warning is returned when the precision of a value may be lost in processing OMF Data messages.

Bug Fixes:

- None.

PI Web API OMF Services 2021 SP2

For more information, see the PI Web API OMF Services 2021 SP2 detailed changes topic.

New features:

- Support for per-request response verbosity has been added.
    - For more information, see the PI Web API OMF Services 2021 SP2 detailed changes topic.

Enhancements:

- **The 'OmfIncludeInnerEvents' configuration option has been added. This setting controls whether responses populate the 'InnerEvents' field by default.**
    - Previously, responses always included the 'InnerEvents' field, but it would only be populated when the 'DebugMode' configuration was enabled. Now, the 'InnerEvents' field will be populated based on this setting.
    - Available options are `True` and `False`.
    - The default setting is `True`.
    - For more information, see the PI Web API OMF Services 2021 SP2 detailed changes topic.
- `ContainerOverlapsWithAnotherContainer` **events will now return status code 409 (Conflict), instead of 500 (Internal Server Error).**
- **Improved error handling when connecting to Asset Framework and Data Archive servers.**
    - Previously, when an Asset Framework or Data Archive server was not found or could not be connected to, a 500 (Internal Server Error) would be returned.
    - Now, a 503 (Service Unavailable) will be returned.
- **Improved error handling when an operation fails due to a network interruption to Asset Framework or Data Archive.**

  - Previously, when an operation failed due to a network interruption, a generic 500 (Internal Server Error) would be returned.
  - Now, in most cases, a 504 (Gateway Unavailable) will be returned. Some operations which can gracefully recover from a network interruption may return a more specific status code if data was lost as a result of the network interruption.
- Some operations will return additional parameters indicating what `type` or `container` the event is related to.
  - Previously, the client needed to check the message index to determine which resource caused the failure. Now, the client can choose to rely on the event parameters if they are available and using the message index is too cumbersome.
    - Note that the set of parameters returned by an operation may change between OMF versions or releases of the PI Web API. When possible, the message index is still the preferred mechanism to determine which resource caused a failure.
- The `PointReferenceAttributesUpdated` metric now reflects the number of AF Attributes updated when processing a `link`.
  - Previously, these attributes were rolled up under the `PointReferenceAttributesCreated` metric.

Bug Fixes:

- `update data` operations that include `link` messages no longer incorrectly report mismatches when a `link` already exists.
  - Previously, the mismatch would prevent the `link` from being updated. Now, the `link` will be correctly updated.
- Operations involving static-to-dynamic `links` where the target container includes an enumeration property will no longer incorrectly report PI Point mismatches.
  - Previously, a `create data` request that included a static-to-dynamic `link` where the target container included an enumeration property would succeed, but subsequent operations on the LINK (ex. re-submitted `create` operations, `delete` operations, etc.) would incorrectly report a PI Point mismatch. Now, the check is performed correctly, and subsequent operations will succeed (or fail with the correct error message).

PI Web API OMF Services 2021 SP1

For more information, see the PI Web API OMF Services 2021 SP1 detailed changes topic.

New features:

- Extended support for version 1.2 of the OMF specification has been added.
  - For more information, see the PI Web API OMF Services 2021 SP1 detailed changes topic.

Enhancements:

- Performance improvements for `create` and `update container` operations.
- Performance improvements for `data` operations.
- Some events have been renamed.
  - For more information, see the PI Web API OMF Services 2021 SP1 detailed changes topic.

Bug Fixes:

- **The interpolation mode of PI Points created using earlier versions of PI Web API is now ignored.**
  - Full interpolation mode support is available as of this release. Previously, interpolation mode was an internal feature of PI Web API and could not be controlled.
  - For more information, see the PI Web API OMF Services 2021 SP1 detailed changes topic.
- Container messages that include unsupported property overrides will now return accurate events.
  - Previously, when a container message included property overrides for a reference property, if a property override was not supported, the event would incorrectly identify which override was not supported. Now, the event parameters will correctly indicate which property overrides are not supported.
- Minimum and maximum values derived from PI Point Span and Zero attributes are now clamped to the range of the specified property type and format.
  - Previously, if an OMF PI Point's Span and Zero attributes represented minimum or maximum values outside the range of the specified property type format, some operations that consume the PI Point could fail (such as creating a link). Now, the minimum and maximum values will be clamped to the allowed range, and the operation will not fail.
- Links can be created when a PI Point has an Engineering Unit that does not correspond to an AF UOM.
  - Previously, if an OMF PI Point's Engineering Unit attribute did not correspond to an AF UOM, the link operation would fail. Now, the link will be created without an associated UOM.

PI Web API OMF Services 2021

For more information, see the PI Web API OMF Services 2021 detailed changes topic.

New features:

- Partial support for version 1.2 of the OMF specification has been added.
  - OMF version 1.2 introduces many new features, including:
    - Enumeration types
    - Property minimum and maximum support
    - Overriding type-level information when processing containers
  - For more information, see the following resources:
    - OMF v1.2 Specification, for more about OMF v1.2.
    - PI Web API OMF Services 2021 detailed changes, for more about the features supported in this release of PI Web API.
    - PI Web API Companion Guide, for more on PI Web API specific OMF development considerations and tips.
- The `update` action is now partially supported for the `container` resource type.
  - For more information, see the PI Web API OMF Services 2021 detailed changes topic.
- Improved PI Point migration support is now available.
  - For more information, see the PI Web API OMF Migration topic.
- Performance metrics for OMF are now available.
  - For more information, see the Performance Metrics - OMF topic.

Enhancements:

- **Some event info parameter names have been changed for readability.**
- `array`, `object`, **and non-**`Date-Time string` **properties are now implicitly nullable.**

- `UpdateOperationCreatedNewResource` **has been replaced with more specific events.**
  - Depending on the type of resource created, one of the following events will be returned:
    - `ContainerCreated`
    - `LinkCreated`
    - `StaticInstanceCreated`
- **Some events regarding resource collisions have been updated.**
  - **Operations that previously returned** `TypeMismatch` **now return** `TypeConflictsWithExisting`**.**
  - For more information, see the PI Web API OMF Services 2021 detailed changes topic.
- The 'OmfCreateMode' configuration option has been added. This setting indicates how to treat duplicate values when writing `dynamic data` to the Data Archive.
  - Available options are `Replace`, `Insert`, and `NoReplace`.
  - The default mode is `Insert`.
- Existing `digital` PI Points can now be adopted by containers.
- The per-user OMF cache can now be cleared using the `Cache-Control: no-cache` header.
  - For more information, see the PI Web API OMF Services 2021 detailed changes topic.
- Dynamic data for `int32` properties will no longer return `FeatureNotSupported` for very large negative numbers.
  - Previously, these values were rejected because the Data Archive cannot faithfully store them. Now, the value will be coerced to the `Over Range` Digital State.

Bug Fixes:

- **Containers using a type that has no non-system properties will be rejected.**
  - Previously, containers that used a type that had only system properties would be accepted, but no PI Points would be created. This meant container IDs could be incorrectly re-used.
- `StaticDataDoesNotMatchExisting` **events will now return status code 409 (Conflict), instead of 400 (Bad Request).**
- `delete container` **messages that do not match the existing container will now return status code 409 (Conflict), instead of 200 (OK).**
  - Previously, if a `delete container` message did not match the existing container, PI Points would not be deleted, but a 200 (OK) would still be returned. Although the underlying behavior has not changed, clients may need to be updated to account for the new status code.
- Types containing properties which specify a UOM may now be deleted using either the UOM's full name, or its abbreviation.
- A PI Point extended descriptor that ends in a comma will no longer cause requests to hang.
- A `delete type` message that is rejected due to a mismatch between the message and the existing resource now includes information about the detected mismatch.
- Static-to-dynamic links now set attribute UOMs.

PI Web API OMF Services 2019 SP1

For more information, see the PI Web API OMF Services 2019 SP1 detailed changes topic.

New features:

- Limited support for the `update` action has been added.

  - `update` is now supported for `data` requests for the `static`, `dynamic`, and `__link` resource types.
  - For more information, see the OMF Endpoint Notes topic.
- **Naming conventions have been changed for some resource types.**
  - For more information, see the PI Web API OMF Migration topic.
- **Limited migration support has been added for some resource types.**
  - For more information, see the PI Web API OMF Migration topic.

Fixes and Enhancements:

- `ContainerNotFound` **and** `StaticInstanceNotFound` **events will now return status code 404 (Not Found), instead of 400 (Bad Request).**
- `TypeDoesNotHavePropertyWithSpecifiedIndex` **has had its severity downgraded from** `Error` **to** `Info`**.**
- The string values `"NaN"`, `"Infinity"`, and `"-Infinity"` are now accepted for properties of type `number`.
- Bug fixes

PI Web API OMF Services 2019

For more information, see the PI Web API OMF Services 2019 detailed changes topic.

New features:

- OMF v1.1 data ingress through PI Web API endpoint now supported over HTTPS when Open Message Format Services are installed
  - Support for `create` and `delete` actions.
  - `update` actions are not implemented.
  - For more information, see the OMF Endpoint Notes topic.

Analysis

An Analysis is used to execute an analysis on a set of data values for an Element, based on some analysis rules and time rule.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetAnalysesQuery
- GetCategories
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

AnalysisCategory

An analysis category represents a user-defined value used to categorize analyses and analysis templates.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

AnalysisRule

An Analysis Rule defines the information needed for an analysis of the data within a case, and executes that analysis. Analysis Rules can be applied to Analyses and Analysis Templates.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateAnalysisRule
- GetAnalysisRules

AnalysisRulePlugIn

An Analysis Rule Plug-in stores its information in an Asset Server, and can be used to create an Analysis Rule for an Analysis or an Analysis Template.

Actions

- Get
- GetByPath

AnalysisTemplate

An analysis template is used to create a stored definition of an analysis, and can be used to create analyses.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateFromAnalysis
- CreateSecurityEntry
- DeleteSecurityEntry
- GetAnalysisTemplatesQuery
- GetCategories
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

AssetDatabase

An Asset Database stores its information in an Asset Server, and provides access to basic entities represented on each database such as elements, element templates, and event frames.

Actions

- Get
- GetByPath
- Update
- Delete
- AddReferencedElement
- CreateAnalysisCategory
- CreateAnalysisTemplate
- CreateAttributeCategory
- CreateElement
- CreateElementCategory
- CreateElementTemplate
- CreateEnumerationSet
- CreateEventFrame
- CreateSecurityEntry
- CreateTable
- CreateTableCategory
- DeleteSecurityEntry
- Export
- FindAnalyses
- FindElementAttributes
- FindEventFrameAttributes
- GetAnalysisCategories
- GetAnalysisTemplates
- GetAttributeCategories
- GetElementCategories
- GetElements
- GetElementTemplates
- GetEnumerationSets
- GetEventFrames
- GetReferencedElements
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- GetTableCategories
- GetTables
- Import
- RemoveReferencedElement
- UpdateSecurityEntry

AssetServer

An Asset Server represents a single logical data store. The server includes references to individual Asset Databases contained within it, as well as supported units of measure.

Actions

- List
- Get
- GetByPath
- CreateAssetDatabase
- CreateNotificationContactTemplate
- CreateSecurityEntry
- CreateSecurityIdentity
- CreateSecurityMapping
- CreateUnitClass
- DeleteSecurityEntry
- GetAnalysisRulePlugIns
- GetByName
- GetDatabases
- GetNotificationContactTemplates
- GetNotificationPlugIns
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- GetSecurityIdentities
- GetSecurityIdentitiesForUser
- GetSecurityMappings
- GetTimeRulePlugIns
- GetUnitClasses
- UpdateSecurityEntry

Attribute

An attribute represents a single value that is used to represent a specific piece of information that is part of an element base. Attributes can be applied to elements, element templates, and event frames. An attribute may be based on a template, which will derive its characteristics from the template definition.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateAttribute
- CreateConfig
- GetAttributes
- GetAttributesQuery
- GetCategories
- GetMultiple
- GetValue

- SetValue

AttributeCategory

An attribute category represents a user-defined value used to categorize attributes and attribute templates.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

AttributeTemplate

An attribute template is used to create a stored definition of an attribute. Attribute templates can be used to create common attribute structures for elements and event frames.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateAttributeTemplate
- GetAttributeTemplates
- GetCategories

AttributeTrait

A trait represents an attribute trait that can be assigned to an attribute template or attribute, to define a trait with a well-known relationship to other attributes. See Attribute Trait for more information.

Actions

- Get
- GetByCategory

Batch

PI Web API supports batching multiple logical REST requests into a single HTTP request.

Actions

- Execute

Calculation

These methods provide mechanisms to evaluate Performance Equation expressions over objects in Asset and Data Servers. Calculations over values interpolated at intervals, over recorded values, and at specific times, as well as summary calculations, are supported.

Actions

- GetAtIntervals
- GetAtRecorded
- GetAtTimes
- GetSummary

Channel

A channel is a way to receive continuous updates about a stream or stream set. See Channels for more information.

Actions

- Instances

Configuration

PI Web API contains a configuration store to manage runtime aspects of the system.

Actions

- DeleteConfiguration
- GetConfiguration
- ListActiveConfiguration
- ListConfiguration
- PutConfiguration

DataServer

An Data Server is used to track the data recorded from PI points for a specific process or entity.

Actions

- List
- Get
- GetByPath
- CreateEnumerationSet
- CreatePoint
- GetByName
- GetEnumerationSets
- GetLicense
- GetPoints

Element

An element is a logical grouping of attributes and child elements. An element may be based on a template, which will derive its characteristics from the template definition.

Actions

- Get
- GetByPath
- Update
- Delete
- AddReferencedElement
- CreateAnalysis
- CreateAttribute
- CreateConfig
- CreateElement
- CreateNotificationRule
- CreateSearchByAttribute
- CreateSecurityEntry
- DeleteSecurityEntry
- ExecuteSearchByAttribute
- FindElementAttributes
- GetAnalyses
- GetAttributes
- GetCategories
- GetElements
- GetElementsQuery
- GetEventFrames
- GetMultiple

- GetNotificationRules
- GetPaths
- GetReferencedElements
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- RemoveReferencedElement
- UpdateSecurityEntry

ElementCategory

An element category represents a user-defined value used to categorize elements, event frames, and element templates.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

ElementTemplate

An element template is used to create a stored definition of an element or event frame, which may contain a collection of attributes. Element templates can be used to create element structures for elements or event frames.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateAttributeTemplate
- CreateNotificationRuleTemplate
- CreateSecurityEntry
- DeleteSecurityEntry
- GetAnalysisTemplates
- GetAttributeTemplates

- GetBaseElementTemplates
- GetCategories
- GetDerivedElementTemplates
- GetNotificationRuleTemplates
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

EnumerationSet

An enumeration set represents a user-defined set of named constant enumeration values. Enumeration sets may be used to define the type of an attribute

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- CreateValue
- DeleteSecurityEntry
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- GetValues
- UpdateSecurityEntry

EnumerationValue

An enumeration value represents a user-defined name and value pair contained with an enumeration set

Actions

- Get
- GetByPath
- DeleteEnumerationValue
- UpdateEnumerationValue

EventFrame

Event frames are representations of objects that frame an event with a start and end time. Event frames may be used by elements in order to capture, track, compare, or analyze business events and their related data for a

repeatable time period. An event frame may be based on a template, which will derive its characteristics from the template definition. See Event Frame for more information.

Actions

- Get
- GetByPath
- Update
- Delete
- Acknowledge
- CaptureValues
- CreateAnnotation
- CreateAnnotationAttachmentMediaById
- CreateAttribute
- CreateConfig
- CreateEventFrame
- CreateSearchByAttribute
- CreateSecurityEntry
- DeleteAnnotation
- DeleteAnnotationAttachmentMediaById
- DeleteSecurityEntry
- ExecuteSearchByAttribute
- FindEventFrameAttributes
- GetAnnotationAttachmentMediaById
- GetAnnotationAttachmentMediaDataById
- GetAnnotationAttachmentMediaMetadataById
- GetAnnotationById
- GetAnnotations
- GetAttributes
- GetCategories
- GetEventFrames
- GetEventFramesQuery
- GetMultiple
- GetReferencedElements
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateAnnotation
- UpdateSecurityEntry

Home

The Home page is the highest level of the PI Web API organizational hierarchy. Users can use this page to navigate to all entities within PI Web API via supplied links.

Actions

- Get

Metrics

Metrics represent information related to PI Web API's operations.

Actions

- Environment
- Landing
- Requests

NotificationContactTemplate

A notification contact template represents a contact that can be used to subscribe to a notification rule.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetNotificationContactTemplates
- GetNotificationContactTemplatesQuery
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

NotificationPlugIn

A Notification Plug-in represents the information needed for a notification to receive events, and can be used to create Notification Contact Templates and Notification Rule Subscribers.

Actions

- Get
- GetByPath

NotificationRule

A notification rule represents objects used to generate a notification.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateNotificationRuleSubscriber
- CreateSecurityEntry
- DeleteSecurityEntry
- GetNotificationRulesQuery
- GetNotificationRuleSubscribers
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

NotificationRuleSubscriber

A notification rule subscriber defines the information needed to deliver notifications.

Actions

- Get
- GetByPath
- Update
- Delete
- GetNotificationRuleSubscribers

NotificationRuleTemplate

A notification rule template is used to create a stored definition of a notification rule, and can be used to create notification rules.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateNotificationRuleTemplateSubscriber

- CreateSecurityEntry
- DeleteSecurityEntry
- GetNotificationRuleTemplatesQuery
- GetNotificationRuleTemplateSubscribers
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

Omf

OMF plugin controller.

Actions

- PostAsync

Point

A PI Point is a configured item in a Data Server which tracks the value for a specific piece of data. PI Points are used for representing the value or status of a given process during a specific point in time or for a specific instance in the process.

Actions

- Get
- GetByPath
- Update
- Delete
- GetAttributeByName
- GetAttributes
- GetMultiple
- GetPointsBySearch
- UpdateAttributeValue

SecurityIdentity

A Security Identity represents the identity of an authenticated user on an Asset Server. Once authenticated, access permissions on resources define authorization against the user's Security Identity.

Actions

- Get
- GetByPath

- Update
- Delete
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- GetSecurityMappings

SecurityMapping

A Security Mapping represents the relationship between a Security Identity and a Windows identity on an Asset Server.

Actions

- Get
- GetByPath
- Update
- Delete
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName

Stream

PI Web API supports retrieving time series data in various formats, such as interpolated, plot, recorded, and summary values. In addition, updating the current value is supported. See Stream for more information.

Actions

- GetChannel
- GetEnd
- GetInterpolated
- GetInterpolatedAtTimes
- GetPlot
- GetRecorded
- GetRecordedAtTime
- GetRecordedAtTimes
- GetSummary
- GetValue
- RegisterStreamUpdate
- RetrieveStreamUpdate
- UpdateValue
- UpdateValues

StreamSet

The PI Web API supports bulk data retrieval using stream sets. A stream set is a set of stream attributes with a common parent element, event frame, or attribute. Specify the WebID of the parent together with various filters to query multiple data references for interpolated, plot, recorded, summary, or current values simultaneously. For the AdHoc actions, only specify WebId of a stream. See Stream for more information.

Actions

- GetChannel
- GetChannelAdHoc
- GetEnd
- GetEndAdHoc
- GetInterpolated
- GetInterpolatedAdHoc
- GetInterpolatedAtTimes
- GetInterpolatedAtTimesAdHoc
- GetJoined
- GetPlot
- GetPlotAdHoc
- GetRecorded
- GetRecordedAdHoc
- GetRecordedAtTime
- GetRecordedAtTimeAdHoc
- GetRecordedAtTimes
- GetRecordedAtTimesAdHoc
- GetSummaries
- GetSummariesAdHoc
- GetValues
- GetValuesAdHoc
- RegisterStreamSetUpdates
- RetrieveStreamSetUpdates
- UpdateValue
- UpdateValueAdHoc
- UpdateValues
- UpdateValuesAdHoc

System

The System information provides access to PI Web API configuration, cached instances, current user and version information.

Actions

- CacheInstances

- Landing
- RemoveCacheInstance
- Status
- UserInfo
- Versions

Table

A table stores user-defined information in a .NET DataTable.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetCategories
- GetData
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateData
- UpdateSecurityEntry

TableCategory

A table category represents a user-defined value used to categorize tables.

Actions

- Get
- GetByPath
- Update
- Delete
- CreateSecurityEntry
- DeleteSecurityEntry
- GetSecurity
- GetSecurityEntries
- GetSecurityEntryByName
- UpdateSecurityEntry

TimeRule

The Time Rule provides a way to create case time periods based on user supplied logic and configuration. Time Rules can be applied to Analyses and Analysis Templates.

Actions

- Get
- GetByPath
- Update
- Delete

TimeRulePlugIn

A Time Rule Plug-in stores its information in an Asset Server, and can be used to create a Time Rule for an Analysis or an Analysis Template.

Actions

- Get
- GetByPath

Unit

The unit describes a granular specification for the unit of measure. For example, a unit "meter" may exist in an unit of measure "length".

Actions

- Get
- GetByPath
- Update
- Delete

UnitClass

Units of measure define a standard description to express a quantity or value. Each unit of measure contains granular units to describe the unit of measure. For example, a unit of measure may be "length", which contains units "meter" and "kilometer".

Actions

- Get

- GetByPath
- Update
- Delete
- CreateUnit
- GetCanonicalUnit
- GetUnits

Associations (Core Services)

Associations modify requests to return more information. Usually this requires extra calls to the AF or PI Data Archive servers so it is omitted by default. The action documentation indicates what associations it supports. The following associations are accepted:

- **Annotations** Specifies to return the annotations associated with the AF value returned from a data request.
- **Paths** Specifies to return all the full paths associated with the AF element(s) returned from a request.

Attribute Trait (Core Services)

Represents the attribute trait assigned to an attribute template or attribute to define a trait with a well-known relationship to other attributes. The trait can be used when referencing an attribute by path instead of needing to know the name of the attribute. For example, the path "|Pressure|[@Trait=LoLo]" would reference attribute with the LimitLoLo trait under the "Pressure" attribute. Traits may also have well-known behaviors. For example, attributes with limit traits will have the same Type and DefaultUOM as the parent attribute.

Attribute traits are divided into six categories: **Analysis**, **Forecast**, **Health**, **Limit**, **Location** and **Other**. Names, categories and abbreviations of all attribute traits are:

Attribute Trait Name	Category	Abbreviation
AnalysisStartTriggerExpression	Analysis	StartTriggerExpression
AnalysisStartTriggerName	Analysis	StartTriggerName
Forecast	Forecast, Other	Forecast
HealthScore	Health	Score
HealthStatus	Health	Status
LimitHi	Limit	Hi
LimitHiHi	Limit	HiHi
LimitLo	Limit	Lo
LimitLoLo	Limit	LoLo
LimitMaximum	Limit	Maximum
LimitMinimum	Limit	Minimum
LimitTarget	Limit	Target
LocationAltitude	Location	Altitude
LocationLatitude	Location	Latitude
LocationLongitude	Location	Longitude
Reason	Other	Reason

Boundary Type (Core Services)

Defines the behavior of data retrieval at the end points of a specified time range. The following values are accepted:

- Inside

  Specifies to return the recorded values on the inside of the requested time range as the first and last values. When used as the boundary type for a sync time, all returned values are guaranteed to be within the specified time range.

- Outside

  Specifies to return the recorded values on the outside of the requested time range as the first and last values. When used as the boundary type for a sync time, the returned values can be outside the specified time range by no more than one interval. For more details, see the Sync Time help topic.

- Interpolated

  Specifies to create an interpolated value at the end points of the requested time range if a recorded value does not exist at that time. This boundary type is not supported for sync times.

Buffer Option (Core Services)

Indicates how to buffer value updates.

- DoNotBuffer

  Update values without buffering.

- BufferIfPossible

  Attempt to update with buffering. If this fails, fall back to updating without buffering.

- Buffer

  Update values with buffering.

Calculation Basis (Core Services)

Defines the possible calculation options when performing summary calculations over time-series data. The following values are accepted:

- TimeWeighted

  Weight the values in the calculation by the time over which they apply. Interpolation is based on whether the attribute is stepped. Interpolated events are generated at the boundaries if necessary.

- EventWeighted

  Evaluate values with equal weighting for each event. No interpolation is done except in the case of expression calculations evaluated by the PI Server. There must be at least one event within the time range to perform a successful calculation. Two events are required for standard deviation. In handling events at the boundary of the calculation, the AF SDK uses following rules:

  a. use events at both boundaries when there is only one calculation interval;
  b. include events at start time in multiple intervals and the intervals are in ascending time order;
  c. include events at the end time in multiple intervals and the intervals are in descending time order.

- TimeWeightedContinuous

  Apply weighting as in _TimeWeighted_, but do all interpolation between values as if they represent continuous data, (standard interpolation) regardless of whether the attribute is stepped.

- TimeWeightedDiscrete

Apply weighting as in *TimeWeighted* but interpolation between values is performed as if they represent discrete, unrelated values (stair step plot) regardless of the attribute is stepped.

- EventWeightedExcludeMostRecentEvent

  The calculation behaves the same as _EventWeighted_, except in the handling of events at the boundary of summary intervals in a multiple intervals calculation. Use this option to prevent events at the intervals boundary from being double count at both intervals. With this option, events at the end time (most recent time) of an interval is not used in that interval.

- EventWeightedExcludeEarliestEvent

  Similar to the option _EventWeightedExcludeMostRecentEvent_. Events at the start time(earliest time) of an interval is not used in that interval.

- EventWeightedIncludeBothEnds

  Events at both ends of the interval boundaries are included in the event weighted calculation.

Channels (Core Services)

A channel is a way to receive continuous updates about a stream or stream set. Rather than using a typical HTTP request, channels are accessed using the Web Socket protocol. There are numerous Web Socket client libraries in several languages available for use with channels.

Channels use the "wss://" scheme instead of "https://" to indicate the use of secure Web Sockets. Below are some examples of channel URLs:

- wss://myserver/piwebapi/streams/{webId}/channel
- wss://myserver/piwebapi/streamsets/{webId}/channel
- wss://myserver/piwebapi/streamsets/channel?webId={webId}

Once connected, the server will send messages containing all the stream value changes since the last message. The payload of these messages is the same as the "GetEnd" method for streams and stream sets. Any streams whose values have not changed since the last message will not be included in the message. Value changes are not guaranteed to be sent in chronological order.

Below is an example of a channel message:

```
{
    "Links": {},
    "Items": [
        {
            "WebId": "I1AbEDqD5loBNH0erqeqJodtALAaqQoQHk26BGgMQAVXYR0Ag8rv
bPp_xUEC80vMunHcq2g",
            "Name": "Water",
            "Links": {
                "Self": "https://myserver/piwebapi/attributes/I1AbEDqD5loB
NH0erqeqJodtALAaqQoQHk26BGgMQAVXYR0Ag8rvbPp_xUEC80vMunHcq2g"
            },
            "UnitsAbbreviation": "m",
            "Items": [
                {
```

```
                    "Timestamp": "2014-07-22T14:00:00Z",
                    "Value": 2.3,
                    "Good": "true",
                    "Questionable": "false",
                    "Substituted": "false"
                },
                {
                    "Timestamp": "2014-07-22T15:00:00Z",
                    "Value": 2.5,
                    "Good": "true",
                    "Questionable": "false",
                    "Substituted": "false"
                }
            ]
        }
    ]
}
```

For PI Points and AF Attributes with PI Point Data Reference, channels will also send messages about modifications or deletions of previous values. Modified values will have the "Substituted" flag set to true, and deleted values will have the "Good" flag set to false and a digital state of "No Data".

Error Handling

While retrieving data, if there is a communication error with a data source, errors are emitted similar to the following:

```
{
    "Errors": [
        "An exception occurred while retrieving values through Data Server
: 'MyPIServer'. Exception Message: '[-10723] PINET: No Connection.'"
    ]
}
```

For more information see the Error Handling topic.

Polling Interval

During operation, the Channel will periodically poll the PI System's data update queue. The interval between these polls is called the "polling interval". If there are any data updates detected while polling, PI Web API will send them to the client in a message. The polling interval can be changed using the ChannelPollingInterval configuration value. The units are in milliseconds and the default value is 1000.

Query Parameters

If the URL parameter "includeInitialValues" is set to true, the server will send an initial message containing the current values of all the streams. Below is an example:

- wss://myserver/piwebapi/streams/{webId}/channel?includeInitialValues=true

If the URL parameter "heartbeatRate" is set, the Channel will periodically send an empty message to confirm that the connection is still alive. The "heartbeatRate" parameter is specified as an integer multiple of the polling interval. After that many polling intervals without a data update, an empty message will be sent. By default, no empty messages will be sent to the client.

Below is an example:

- wss://myserver/piwebapi/streams/{webId}/channel?heartbeatRate=5

In this case, an empty message will be sent after every five polling intervals without a data update.

Sample Clients

Below are some sample clients for channels:

Sample JavaScript Client (Requires Browser with support for Web Sockets)

```
var uri = "wss://myserver/piwebapi/streams/{webId}/channel";

var webSocket = new WebSocket(uri);

webSocket.onopen = function(event)
{
    console.log("Connection opened.");
};

webSocket.onerror = function(event)
{
    console.log("Connection aborted.");
};

webSocket.onclose = function(event)
{
    console.log("Connection closed.");
};

webSocket.onmessage = function(event)
{
    console.log("Message received: " + event.data);
};
```

Sample C# Client (Requires .NET 4.0 or newer)

```
using System;
using System.Net.WebSockets;
```

```
using System.Text;
using System.Threading;
using System.Threading.Tasks;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Press any key to close.");

        CancellationTokenSource cancellationSource = new CancellationToken
Source();

        Task runTask = RunClient(cancellationSource.Token);

        Console.ReadKey();

        cancellationSource.Cancel();

        runTask.Wait();
    }

    public static async Task RunClient(CancellationToken cancellationToken
)
    {
        Uri uri = new Uri("wss://myserver/piwebapi/streams/{webId}/channel
");

        WebSocketReceiveResult receiveResult;
        byte[] receiveBuffer = new byte[65536];
        ArraySegment<byte> receiveSegment = new ArraySegment<byte>(receive
Buffer);

        using (ClientWebSocket webSocket = new ClientWebSocket())
        {
            try
            {
                await webSocket.ConnectAsync(uri, CancellationToken.None);
            }
            catch (WebSocketException)
            {
                Console.WriteLine("Could not connect to server.");
                return;
            }

            while (true)
            {
                try
```

```
                {
                    receiveResult = await webSocket.ReceiveAsync(receiveSe
gment, cancellationToken);
                }
                catch (OperationCanceledException)
                {
                    break;
                }

                if (receiveResult.MessageType != WebSocketMessageType.Text
)
                {
                    await webSocket.CloseAsync(
                        WebSocketCloseStatus.InvalidMessageType,
                        "Message type is not text.",
                        CancellationToken.None);
                    return;
                }
                else if (receiveResult.Count > receiveBuffer.Length)
                {
                    await webSocket.CloseAsync(
                        WebSocketCloseStatus.InvalidPayloadData,
                        "Message is too long.",
                        CancellationToken.None);
                    return;
                }

                string message = Encoding.UTF8.GetString(receiveBuffer, 0,
 receiveResult.Count);

                Console.WriteLine(message);
            }

            await webSocket.CloseAsync(
                WebSocketCloseStatus.NormalClosure,
                "Closing connection.",
                CancellationToken.None);
        }
    }
}
```

Data Server Licenses (Core Services)

The Data Server contains license information for various types of licensed items. Licensed items represent properties such as whether an application is allowed to connect, or the number of allowed data streams. License limits are specific to an individual Data Server. Each Data Server has its own license database.

The PI Web API refers to individual licensed items as `Modules`. Modules have unique, case-sensitive names.

The PI Web API requires that modules be explicitly specified by name. It is not possible to enumerate the modules on a Data Server through the PI Web API.

Please refer to OSIsoft Tech Support Knowledge Base Article 3157OSI8. The PI Web API also exposes information about data stream use, organized by module. This allows for the ability to determine how many data streams are being used by a given component or module, and how many remain available. The overall allowed point count of a PI Server can be determined using the `pibasess.MaxAggregatePointModuleCount` and `pibasess.maxpointcount` modules. Of these two modules, the module with the lower `Total` field indicates the maximum point count.

Element Type (Core Services)

The possible values for the type of an element. The following values are accepted:

- None

  The element type is not defined. This is the default value for new element templates and elements without a template.
- Other

  The element type is something other than one of the system-defined types. This type is to be used when one of the other system-defined types is not adequate.
- Node

  The type of the element is a Node.
- Measurement

  The type of the element is a Measurement. A measurement is normally used for an element which is used to measure values of another element (i.e. a meter).
- Flow

  The type of the element is a Flow. A flow is normally used to represent the flow of material or information between two elements.
- Transfer

  The type of the element is a special kind of a time-based flow which is called a Transfer.
- Boundary

  The type of the element is a Boundary. Elements of this type do not usually participate in any model analysis and are considered boundaries to the model.
- PIPoint

  The type of the element is a point. Elements of this type are used to represent a point in a Data Server as an element. Users should not create elements with this type.
- Any

  The allowed element type for AllowedElementTypes can be any of the defined types. This type is only allowed when setting the allowed element types for connection port. This type cannot be used when setting the type of an element.

Event Frame (Core Services)

Each event frame has a name, start time, end time, one or more attributes, and one or more referenced AF elements. An event frame shares some similarities with an element but contains additional metadata. You can perform many event frame actions with specific element-related controllers.

- Event Frame Template

  As with element templates, you create event frame templates to standardize and manage the attributes for different types of events, rather than a specific asset. Because event frame templates are a special type of element template, you can perform many of the same actions:

  - To create an event frame template, CreateElementTemplate in the asset database and specify the InstanceType to be "EventFrame".
  - To view the list of event frame templates in an asset database, GetElementTemplates in the asset database. This action retrieves both element and event frame templates.
  - To view, update, or delete individual event frame templates, you can use any available actions in element template.
- Event Frame Category

  An event frame category is equivalent to an element category.
- Event Frame Attribute and Attribute Template

  You access attributes that are defined in event frames in attribute, whereas you access attribute templates defined in event frame templates in attribute template.

Export Mode (Core Services)

The export mode indicates the type of export to perform when exporting a database. The following values are accepted:

- StrongReferences

  Export this object and all objects it contains.
- AllReferences

  Export this object and all objects it references. This flag cannot be set if the NoReferences flag is set.
- NoReferences

  Export this object only. For example, AF Attributes of an AF Element will not be included. This flag cannot be set if the AllReferences flag is set.
- NoUniqueID

  Don't export UniqueIDs associated with each object.
- DefaultValues

  Export default values.
- Library

  Used to limit the export of a database to include library objects only. Library objects are Categories, all Templates, Enumeration Sets, Reference Types, Tables and the Unit-of-Measure Database.
- Security

Include Security Descriptors in export. Exporting Security Descriptors will have a performance impact on both Export and Import.

- Flat

  Export hierarchical objects in a flattened style. This includes Elements, Attributes, Attribute Templates, and Event Frames.

- SimplifiedConfigStrings

  Export the simplified form of configuration strings for attributes and attribute templates. The simplified strings will already have parameters substituted and will not include UniqueIDs or point identifiers.

- AppendUnitsOfMeasure

  Append the unit of measure, if any, to the end of attribute values and attribute template values when they are exported.

- PasteOperation

  Use user interface Paste Rules to determine what properties of templated objects should be exported.

Import Mode (Core Services)

The import mode indicates the type of import to perform when importing a database. The following values are accepted:

- AllowCreate

  Import and create new objects.

- AllowUpdate

  Import and overwrite existing objects. This flag cannot be set if the GenerateUniqueNames flag is set.

- GenerateUniqueNames

  Generate unique names for objects that conflict. This flag cannot be set if the AllowUpdate flag is set. If this flag set, then the AllowCreate flag must also be set.

- PasteOperation

  Use user interface Paste Rules to determine when to generate unique names or allow updates. You do not usually specify any other flags when setting this flag.

- AutoCheckIn

  Automatically check in all database changes during the import. This can be useful when importing large files which would be too large to check in all changes. If an error occurs during the import, some changes may have already been checked in and cannot be undone.

- Compatibility

  Internal flag set by the compatibility layer to indicate that the import should abide by AF 1.x rules. In particular, importing a global element into a model will create a reference to the element instead of a copy.

- CreateConfig

  Will invoke the Data Reference CreateConfig option, which allows PI Points to be auto created, as well as having their substitution parameters resolved and server and point id's set. Performance of import will be affected when this option is turned on. Note that this setting will only apply to newly created elements for xml import, but will apply to both new and modified elements for csv import.

- DeleteOperation

  Delete existing objects. This flag cannot be set if the AllowCreate or AllowUpdate are set.
- CreateCategories

  Automatically add categories that are specified but not found.

Include Mode (Core Services)

Include mode for the return list.

- All

  Include every item or result into the return list.
- ItemsWithExceptionOnly

  Include only items with exception into the return list.
- ItemsWithoutExceptionOnly

  Include only items without exception into the return list.

Path Syntax (Core Services)

Some PI Web API methods accept AF paths as parameters.

The path is segmented into parts, and each part represents an object or list of objects derived from the AFObject base class. Parts are typically separated with a single backslash (\), with the exception of either an AFAttribute or an AFAttributeTemplate, which use the pipe character (|). Additionally, the PISystem or PIServer portion begins with two backslashes (\\). Depending on the encoding option chosen, each part of the path will contain information on the specific object(s) in that part of the path. That information may consist of the type of collection the object is in, as well either the Name and/or UniqueID of the object or one or more collection filters. There is a default object type for each parent object, therefore the type of collection for the object is only required if it is not the default or a filter is specified. The default collection types are specified in the Default Collection Types table in the AF SDK documentation.

If the collection is specified, then the collection filter or Name and/or UniqueID are enclosed in square brackets ([and]). If both the Name and UniqueID are specified, then they are separated by a semicolon (;), and their relative order defines the precedence when used when finding the object (the first one specified has higher precedence). UniqueIDs must include the surrounding curly brackets ({ and }). A single period enclosed in square brackets ([.]) represents the default collection member of the parent object.

A collection filter starts with the at sign (@) followed by the filter name. Multiple filters may be specified and are evaluated in the specified order. The index filter "[@Index=int]" or "[int]" is used to specify the index of the matched object to return and should be the last filter specified. If the index filter is not specified, then the first match is returned if returning a single object. When using the index filter, the first item is at index 1. The Path Filter table below specifies the supported filters. If a filter does not apply to the type of object in the collection, then an object will not be found.

The PISystem starts a fully qualified path and is preceded by two backslashes (\\). The collection name "Systems" is not needed since its location within the path is well known. A relative path starting with "\\." indicates to begin using the same system as the relative object. For example, "\\.\Database2" can be used to reference a database in the same system as the relative object. If the relative object is not specified or the PISystem is not specified as part of the path, then the PISystems.DefaultPISystem will be used.

The AFDatabase is the default object which follows a PISystem. To access other non-database objects off of the system, the collection must be identified (e.g. "\\MySystem\Contacts[JSmith]"). A relative path starting with a single backslash (\), indicates to begin using the same database as the relative object. For example, paths "\Element2" and "\Tables[MyTable]" can be used to reference objects in the same database as the relative object. If the relative object is not specified or the AFDatabase is not specified as part of the path, then the AFDatabases.DefaultDatabase will be used.

The parent is indicated by a double period (..). An example parent path is "..\Element2|Attribute1". The single period (.) represents the current relative object and can be used to create a relative path from itself. When the relative object is an AFAttribute, then the single period followed by a backslash (.\) represents the owning AFBaseElement while the single period followed by a vertical bar (.|) will reference a child AFAttribute. Examples of a self relative path are ".\|Attribute1" and ".|Attribute1|Attribute2" where the specified relative object was an AFAttribute. All other Element reference types will create a relative path from the database: "\Element1\Element2|Attribute1".

Paths beginning with the AFDatabase (e.g. "\\Database\Element") are **not** valid. In order to specify a path on a database you must specify either a fully qualified path starting with the PISystem (e.g. "\\MySystem\Database\Element"), or a relative path (e.g. "\Element").

For a complete specification of the AF path grammar, see the AF SDK documentation.

Note that when retrieving resources by path, a fully qualified path or a relative path whose defaults imply a fully qualified path is required. Example paths:

\\MySystem\MyDatabase
\\Systems[MySystem]\Databases[MyDatabase]
\\{5c64c379-c182-4f35-8d30-78d8c2f84502};MySystem\{5c64c379-c182-4f35-8d30-78d8c2f84503};MyDatabase
\\MySystem\Databases[MyDatabase]\Elements[@Category=Tutorial]|Volume

Performance Equations (Core Services)

Expression variables are references to attributes or points using relative syntax to the target, which is typically a database, element or a Data Server. Variables must be enclosed in single quotes. Calculations are limited to attributes or points which originate from a single server. Attributes which resolve to a static value (no data reference configured), are also acceptable. Examples of valid target and expressions:

Target	Example Expression
none	'\\myPIServer\sinusoid'*2
none	'\\myAFServer\myDB\myElement|myAttribute'*2
Data Server	'sinusoid'*2
Asset Server	'myRootElement\myChildElement|myAttribute'*2
element	'myAttribute'*2
element	'myAttribute|myChildAttribute'*2
element	'.\myChildElement|myAttribute'*2
element	'\\myPIServer\sinusoid'*2
event frame	'myAttribute'*2
attribute	'.'*2

Target	Example Expression
attribute	'myChildAttribute'*2
attribute	'..|mySiblingAttribute'*2

Note that some characters used in Performance Equations, such as '" and '+', must be URL-encoded.

Performance Metrics (Core Services)

Performance Metrics allows system administrators to view PI Web API usage, system load, and aggregate request performance. These metrics are passively collected by the service and are exposed to PI Web API Administrators through the Metrics controller. Retrieving metrics provides a snapshot of service metrics at the time of the request.

Configuration Settings

Performance Metrics are enabled by default. To disable access to the performance metrics endpoint, set the 'PerformanceMetricsEnabled' configuration option to `false`. This will reset the service's collected metrics. For more information on how to change configuration settings, refer to the Configuration controller or to Configuration at runtime on AVEVA Documentation.

Metrics Collected

Request Metrics

PI Web API Core metrics are organized by HTTP method and route pattern. The PI Web API uses the route pattern to match incoming request URIs to the associated controller and action. URL parameters are ignored during this mapping step, and so they do not appear in Performance Metrics. For example, the request URI */help/controllers/metrics* would appear as */help/controllers/{name}*. Request times are measured in milliseconds. The request time is defined as the time elapsed between when the request is received and when the first byte of the response is returned. This means that the request times measured by the PI Web API do not include external factors, such as the time needed to transmit the response across the network. Core metrics include:

- **Time-bucketed counts** - This set of metrics is designed to act like a histogram. Requests are grouped into buckets based on how long the request took to process. The returned values indicate how many requests were present in each bucket. The label of each bucket is the inclusive lower bound of the range and the next label is its exclusive upper bound. For example, the metrics snippet below indicates that 5 requests completed in 10 or more milliseconds but less than 50 milliseconds, while 3 requests required 120000 or more milliseconds to complete.

```
"Counts": {
  "0": 0,
  "10": 5,
  "50": 0,
  "100": 0,
  "250": 0,
  "500": 0,
```

```
    "750": 0,
    "1000": 0,
    "5000": 0,
    "30000": 0,
    "60000": 0,
    "120000": 3
  }
```

- **Request counts by user**
- **Request counts by status code**
- **Total request count**
- **Last observed time** - The last time a request was observed against this route.
- **Exponentially weighted moving average request time** - This is included as a convenience for users who would like to determine the approximate response time of a given route at a glance. Because it is exponentially weighted, recent requests are more heavily weighted in the average calculation.

Except for the last observed time and average request time, all metrics are expressed as running counts. Thus, to understand the service's performance over a time interval, one may compare a before and after snapshot. Note that Performance Metrics are collected and exposed per PI Web API instance. This means that installations with multiple PI Web API instances behind a load balancer require server affinity to calculate accurate deltas.

Plugin Metrics

The PI Web API consists of several independent groups of functionality, such as Core Services, and OMF Services. These groups are internally referred to as "plugins". Each plugin may collect its own metrics. Plugins may expose zero or more metric counts, as well as zero or more sub-groups of metrics. These sub-groups recursively reuse the same metric count and sub-group structure.

For information regarding the metrics collected by specific plugins, see the below topics. These topics are only available if the corresponding plugin is installed.

- OMF Services Performance Metrics

Environment Metrics

Environment metrics describe the PI Web API host machine.

- **Processor** - Unlike other metrics, the 'Processor' metric is not a request-time snapshot. This metric is periodically sampled and reported as the percentage of CPU in use by PI Web API.
- **Memory** - The current snapshot of private bytes, virtual bytes, and working set bytes in use by PI Web API.

PIPoint Search Query Syntax (Core Services)

PIPoint Search searches within data servers using a query string to define the objects that will be returned by the search. This provides a flexible way to define the criteria used for the search. The syntax used for this query string is described in the Query Syntax section below.

Query Syntax

This is the definition of the syntax used when specifying the search criteria. Search Query syntax described in Extended Backus-Naur Form (EBNF)

```
Query                = AndCondition { "OR" AndCondition } ;

AndCondition         = QueryFilter { AndOperator QueryFilter }
                     | "(" QueryFilter { AndOperator QueryFilter } ")"
                     ;

QueryFilter          = StringValue    (* Defaults to 'Tag:=' [1] *)
                     | PIPOINTATTRIBUTE Operator StringValue [2]
                     | PIPointValueFilter
                     ;

PIPointValueFilter   = "Value" Operator StringValue
                     | "TimeStamp" Operator TimeValue
                     | "Substituted" EqualOperator BooleanValue
                     | "Questionable" EqualOperator BooleanValue
                     | "Annotated" EqualOperator BooleanValue
                     | "IsGood" EqualOperator BooleanValue
                     ;

AndOperator          = "AND" | WHITESPACE ;

Operator             = EqualOperator | ":<>" | ":<" | ":<=" | ":>" | ":>="
;

EqualOperator        = ":" | ":=" ;

TimeValue            = "'" AFTimeString "'"
                     | """ AFTimeString """
                     | AFTimeString
                     ;

BooleanValue         = "'" Boolean "'"
                     | """ Boolean """
                     | Boolean
                     ;

StringValue          = "'" { QuotedEscapedChar | "''" } "'"
                     | """ { QuotedEscapedChar | """" } """
                     | { EscapedChar }
                     ;

QuotedEscapedChar    = Char
                     | "\""      (* Escaped " character *)
```

```
                    | "\'"      (* Escaped ' character *)
                    ;

EscapedChar         = NoWhiteSpaceChar
                    | "\" Char  (* The character is escaped *)
                    ;

Boolean             = "True" | "False" | "1" | "0" ;

AFTimeString        = ? Any string that can be parsed by AFTime.Parse ?

NoWhiteSpaceChar    = ? Any printable character except whitespace characte
rs ?

Char                = ? Any printable character ?
```

```
1.) If a specific filter name is not specified, then the filter will default t
o the
     "Tag" filter and the operator will be "=". When a filter name is specifi
ed, no whitespace
     is allowed between the filter name, the ":" separator, and the optiona
l operator.
     If the operator is not specified, the default operator is "=".
2.) If the type of a PIPOINTATTRIBUTE is DateTime, then the "TimeValue"
     format is supported for the filter value.
```

Wildcard Characters

The string value of a filter can be enclosed in single quotes ('), double quotes ("), or without quotes. Quotations are required if non-escaped white space or quotation marks are desired within the filter string. The filter string value can include regular characters and wildcard characters. Regular characters must match exactly the characters specified in the filter value. Wildcard characters can be matched with arbitrary fragments of the filter value.

When the filter value is specified within either single or double quotes, the single backslash (\) character is treated as a literal character unless followed by a wildcard character, a single quote ('), or a double quote ("). When specified within quotes, two quote characters that match the starting quote character are treated as a single quote character (e.g. '' is treated as a one single quote character ' if the filter value starts with a single quote). When the filter value is specified without quotes, the backslash character is always used to escape the next character. Therefore you must use a double backslash (\\) to match a single backslash when not using quotes around the filter value.

The supported wild card characters are "*" to match any zero or more characters and "?" to match a single character. These characters cannot be escaped using the backslash ("\") character and will always be used as wild card characters within the query value.

Operators

Search operators specify how the filter value is to be compared with the point's value for the filter. For more information about the search operators, see the AFSearchOperator topic.

The following table lists the operators used in the AND condition.

Operator	Description	Example
=	The Equal operator.	Tag:Tank* or Tag:=Tank*
<>	The NotEqual operator.	PointType:<>Int32
<	The LessThan operator.	Value:<100
<=	The LessThanOrEqual operator.	Tag:<=Tank
>	The GreaterThan operator.	Tag:>Tank
>=	The GreaterThanOrEqual operator.	Tag:>=Tank

Caution

Queries with OR condition are not supported for PIPoint value query.

Query Syntax Examples

Below are some example PIPoint queries:

```
query=sin*

query=name:sin*

query=tag:=sin*

tag:<>sin* DataType:Float

tag:<>sin* AND PointType:Float

step:0 AND PointSource:L

(tag:<>sin* AND PointType:Float64) OR (tag:="*Tank*" AND DataType:=Int32)
```

Below are some example of PIPoint value queries, where they can be combined with attribute queries:

Value:=1

This filter would apply to PIPoint value of Numeric, String, and Digital types.

Value:=Auto

This filter would apply to PIPoint value of String and Digital types.

Value:=abc*

This filter would apply to PIPoint value of String type.

Value:="Pt Created"

This filter would apply to PIPoint value of String and PI System Digital State types.

Value:=253 AND IsGood:false

This filter is an alternative to filtering PIPoint value of "Pt Created" PI System Digital State.

Value:=t

This filter would apply to PIPoint value of String and Timestamp types.

tag:sin* AND Value:>10

PointType:Int32 AND Value:>10

PointType:Float32 AND Value:>10

DataType:Float64 AND Value:>1.5

PointType:Blob AND Questionable:true AND TimeStamp:<*

PointType:Blob AND Value:="Pt Created"

PointType:Blob AND Value:=253 AND IsGood:false

PointSource:L AND Annotated:1 AND TimeStamp:t

IsGood:false AND TimeStamp:<"1/5/2017 1:00:00 AM"

ChangeDate:>"2/5/2017 2:00:00 AM" AND Step:0 AND IsGood:1

CreationDate:>y-1d AND Future:true AND TimeStamp:<*

Warnings

Depending on the complexity of the query and the attributes being queried, the action may experience slowness.

The action may cause a decrease in performance for other Data Archive users as well as server-side (Bulk Query limit exceeded) errors, which could impact other user's operations.

For More Information

For more information, see the AF SDK Tech Support article on PIPoint Query Syntax

Reference Type (Core Services)

Reference types define the way in which two elements can relate to one another. The following system defined reference types are available by default:

- Parent-Child

  Defines a 'n..n' strong reference type which will be used to create hierarchical relationships. This is the default reference type when no reference type is specified. This reference type additionally requires unique names.
- Composition

Defines a '1..n' composition reference type which will be used to create compositional hierarchies.

- Weak Reference

  Defines a 'n..n' weak reference type.

Additional reference types that are defined by applications or users can be specified using their names.

Retrieval Mode (Core Services)

The retrieval mode is an enumeration of the possible values for retrieving recorded values from a stream. The following values are accepted:

- Auto

  Automatically determine the best retrieval mode.
- AtOrBefore

  Return a recorded value at the passed time or if no value exists at that time, the previous recorded value.
- Before

  Return the first recorded value before the passed time.
- AtOrAfter

  Return a recorded value at the passed time or if no value exists at that time, the next recorded value.
- After

  Return the first recorded value after the passed time.
- Exact

  Return a recorded value at the passed time or return an error if none exists.

Sample Type (Core Services)

Defines the evaluation of an expression over a time range.

- ExpressionRecordedValues

  Sampling / evaluation is done based on the archive events of all attributes used in the expression. The expression is evaluated at each of the timestamps of these retrieved events. This sampling algorithm is sometimes referred to as natural or event-based.
- Interval

  Sampling / evaluation is done based on a passed expression interval. This is sometimes referred to as evenly-spaced evaluation. This option is only supported for PI Points on PI Data Archive version 3.4 and above.

Search Field (Core Services)

The search field is an enumeration of the fields of the object that are searched. The following values are accepted:

- Name

The object's **Name** property is searched.

- Description

  The object's **Description** property is searched.

- Categories

  The object's **Categories** collection property is searched by name. If only searching categories and the query for the search is null or an empty string, then only items without a category are returned.

- Template

  The object's **Template** property is searched. If only searching the template and the query for the search is null or empty string, then only items without a template are returned.

- Department

  The object's **Department** property is searched.

Search Mode (Core Services)

The search mode is an enumeration of the possible values for specifying the search criteria. The search mode specifies which values are returned relative to the search's start and end times. The following values are accepted:

- None

  This is the value of an uninitialized search mode.

- StartInclusive

  Includes all objects whose start time is within the specified range. Also known as "Starting Between".

- EndInclusive

  Includes all objects whose end time is within the specified range. Also known as "Ending Between".

- Inclusive

  Includes all objects whose start and end time are within the specified range. Also known as "Entirely Between".

- Overlapped

  Includes all objects whose start or end time overlap with the specified range. Also known as "Active Between".

- InProgress

  Includes all objects whose start time is within the specified range and end time is December 31, 9999. Also known as "Starting Between and In Progress".

The following search modes apply only to event frame searches:

- BackwardFromStartTime

  Event frames with a start time less than or equal to the time specified in the search (moving backward in time) are returned from the search. Also known as "Starting Before".

- ForwardFromStartTime

Event frames with a start time greater than or equal to the time specified in the search (moving forward in time) are returned from the search. Also known as "Starting After".

- BackwardFromEndTime

  Event frames with an end time less than or equal to the time specified in the search (moving backward in time) are returned from the search. Also known as "Ending Before".

- ForwardFromEndTime

  Event frames with an end time greater than or equal to the time specified in the search (moving forward in time) are returned from the search. Also know as "Ending After".

- BackwardInProgress

  Event frames with a start time less than or equal to the time specified in the search (moving backward in time) that are still in progress (an end time set to December 31, 9999) are returned from the search. Also known as "Starting Before and In Progress".

- ForwardInProgress

  Event frames with a start time greater than or equal to the time specified in the search (moving forward in time) that are still in progress (an end time set to December 31, 9999) are returned from the search. Also known as "Starting After and In Progress".

Important

Note that object values which end on the search start time or start on the search end time are not included as part of the returned collection.

Search Operator (Core Services)

Search operators for searches.

- Equal

  For "Search by Attribute Value", the attribute value is equal to the value specified in the query. This operator is supported by all attribute value types.

- NotEqual

  For "Search by Attribute Value", the attribute value is not equal to the value specified in the query. This operator is supported by all attribute value types.

- LessThan

  For "Search by Attribute Value", the attribute value is less than the value specified in the query. This operator is supported by all attribute value types other than Boolean, Unsigned Int64, String and Guid.

- GreaterThan

  For "Search by Attribute Value", the attribute value is greater than the value specified in the query. This operator is supported by all attribute value types other than Boolean, Unsigned Int64, String and Guid.

- LessThanOrEqual

  For "Search by Attribute Value", the attribute value is less than or equal to the value specified in the query. This operator is supported by all attribute value types other than Boolean, Unsigned Int64, String and Guid.

- GreaterThanOrEqual

For "Search by Attribute Value", the attribute value is greater than or equal to the value specified in the query. This operator is supported by all attribute value types other than Boolean, Unsigned Int64, String and Guid.

- In

  For "Search by Attribute Value", the attribute value is in the array of the values specified in the query. this operator is supported by all attribute value types other than Single and Double.

Search Option (Core Services)

The **searchOption** parameter specifies how the search endpoint evaluates PI Points according to the supplied tag.

Available options:

Member name	Value	Description
Contains	1	Matches any text that contains the search string.
ExactMatch	2	Matches any text that exactly matches the search string.
StartsWith	3	Matches any text that starts with the search string.
EndsWith	4	Matches any text that ends with the search string.

Search Query Syntax (Core Services)

The AFSearch based searches within the AF SDK use a query string to define the objects that will be returned by the search. This provides a flexible way to define the criteria used for the search. The syntax used for this query string is described in the Search Query Syntax section below.

Query Syntax

This is the definition of the syntax used when specifying the search criteria. Search Query syntax described in Extended Backus-Naur Form (EBNF)

```
Query             = QueryFilter { QueryFilter } ;

QueryFilter       = StringValue   (* Defaults to 'Name:=' (1) *)
                  | "AllDescendants" EqualOperator BooleanValue
                  | "Analysis" EqualOperator StringValue
                  | "AnalysisName" EqualOperator StringValue
                  | "CanBeAcknowledged" EqualOperator BooleanValue
                  | "Category" EqualOperator StringValue
                  | "CategoryName" EqualOperator StringValue
                  | "Contact" EqualOperator StringValue
                  | "ContactName" EqualOperator StringValue
                  | "CreationDate" Operator TimeValue
```

```
                    | "Description" EqualOperator StringValue
                    | "Destination" EqualOperator StringValue
                    | "Duration" Operator TimeSpanValue
                    | "Element" OptionalNestedOrInOperator
                    | "ElementName" EqualOperator StringValue
                    | "ElementReferenceTemplate" EqualOperator StringValue
                    | "End" Operator TimeValue
                    | "EventFrame" NestedQuery
                    | "GroupID" EqualOperator IntegerValue
                    | "ID" EqualOrInOperator
                    | "InProgress" EqualOperator BooleanValue
                    | "IsAcknowledged" EqualOperator BooleanValue
                    | "IsAnnotated" EqualOperator BooleanValue
                    | "IsInternal" EqualOperator BooleanValue
                    | "ModifyDate" Operator TimeValue
                    | "Name" EqualOperator StringValue
                    | "Parent" OptionalNestedQuery
                    | "PlugIn" EqualOperator StringValue
                    | "PlugInName" EqualOperator StringValue
                    | "ReferenceType" EqualOperator StringValue
                    | "Root" EqualOperator StringValue
                    | "Sandbox" EqualOperator BooleanValue
                    | "Severity" Operator SeverityValue
                    | "SortField" Operator SortFieldValue
                    | "SortOrder" Operator SortOrderValue
                    | "Source" EqualOperator StringValue
                    | "Start" Operator TimeValue
                    | "Status" Operator StatusValue
                    | "Target" EqualOperator StringValue
                    | "TargetName" EqualOperator StringValue
                    | "Template" EqualOperator StringValue
                    | "TemplateName" EqualOperator StringValue
                    | "TimeContext" EqualOperator TimeValue
                    | "TimeContextEnd" EqualOperator TimeValue
                    | "Type" EqualOperator StringValue
                    | "Value" EqualOperator AttrValueFilter
                    | AttrValueFilter
                    ;

NestedQuery         = EqualOperator "{" Query "}" ;

OptionalNestedQuery
                    = EqualOperator StringValue
                    | NestedQuery
                    ;

OptionalNestedOrInOperator
                    = OptionalNestedQuery
```

```
                    | InOperator
                    ;

AttrValueFilter     = AttrPath Operator StringValue
                    | AttrPath InOperator
                    | AttrPath Operator StringValue AttrValueType
                    ;

AttrPath            = "'|" { QuotedEscapedChar | "''" } "'"
                    | ""|" { QuotedEscapedChar | """" } """
                    | "|" { EscapedChar }
                    ;

AttrValueType       = As Numeric
                    | As String
                    | As Guid
                    | As AFEnumerationSetName
                    ;

Operator            = EqualOperator | ":<>" | ":<" | ":<=" | ":>" | ":>=" ;

EqualOperator       = ":" | ":=" ;

InOperator          = ":IN(" StringValues ")"

EqualOrInOperator = EqualOperator StringValue | InOperator

SeverityValue       = "None"
                    | "Information"
                    | "Warning"
                    | "Minor"
                    | "Major"
                    | "Critical"
                    ;

SortFieldValue      = "ID"
                    | "Name"
                    | "Type"
                    | "StartTime"
                    | "EndTime"
                    ;

SortOrderValue      = "Ascending"
                    | "Asc"
                    | "Descending"
                    | "Desc"
                    ;
```

```
StatusValue        = "None"
                   | "NotReady"
                   | "Disabled"
                   | "Enabled"
                   | "Error"
                   ;

TimeValue          = "'" AFTimeString "'"
                   | """ AFTimeString """
                   | AFTimeString
                   ;

TimeSpanValue      = "'" AFTimeSpanString "'"
                   | """ AFTimeSpanString """
                   | AFTimeSpanString
                   ;

BooleanValue       = "'" Boolean "'"
                   | """ Boolean """
                   | Boolean
                   ;

IntegerValue       = "'" Integer "'"
                   | """ Integer """
                   | Integer
                   ;

StringValues       = StringValue { ";" StringValue } ;

StringValue        = "'" { QuotedEscapedChar | "''" } "'"
                   | """ { QuotedEscapedChar | """" } """
                   | { EscapedChar }
                   ;

QuotedEscapedChar = Char
                   | "\""      (* Escaped " character *)
                   | "\'"      (* Escaped ' character *)
                   | "\*"      (* Escaped * character *)
                   | "\?"      (* Escaped ? character *)
                   ;

EscapedChar        = NoWhiteSpaceChar
                   | "\" Char  (* The character is escaped *)
                   ;

Boolean            = "True" | "False" | "1" | "0" ;

Integer            = Digit { Digit } ;
```

```
Digit              = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
"9" ;

AFEnumSetName      = ? Any string that matches the name of an AFEnumeration
Set ?

AFTimeString       = ? Any string that can be parsed by AFTime.Parse ?

AFTimeSpanString   = ? Any string that can be parsed by AFTimeSpan.Parse ?

NoWhiteSpaceChar   = ? Any printable character except whitespace characters
 ?

Char               = ? Any printable character ?
```

```
(1) If a specific filter name is not specified, then the filter will default t
o
     "Name" and the operator will be "=". When a filter name is specified, n
o whitespace
     is allowed between the filter name, the ":" separator, and the optiona
l operator.
     If the operator is not specified, the default operator is "=".
```

String values for name filters can contain wildcard characters that are described in the Wildcard Characters section below. The Filters section below describes the filters that can be used in the query and the Operators table below describes the operators that can be used in the filter conditions. The Attribute Value Query section provides more information about searching by attribute values.

Time strings are evaluated in the client's local time unless qualified with time zone information. The time string will be parsed using the current culture. If parsing with the current culture fails, then it will attempt to resolve the time string using the invariant culture.

An empty string value should be used when searching for objects with a null value for the specified filter. For example, to find all objects without a template defined use Template:'' in the query.

Wildcard Characters

The string value of a filter can be enclosed in single quotes ('), double quotes ("), or without quotes. The filter string value can include regular characters and wildcard characters. Regular characters must match exactly the characters specified in the filter value. Wildcard characters can be matched with arbitrary fragments of the filter value. Wildcard characters can be escaped using the single backslash (\) character.

When the filter value is specified within either single or double quotes, the single backslash (\) character is treated as a literal character unless followed by a wildcard character, a single quote ('), or a double quote ("). When specified within quotes, two quote characters that match the starting quote character are treated as a single quote character (e.g. '' is treated as a one single quote character ' if the filter value starts with a single quote). When the filter value is specified without quotes, the backslash character is always used to escape the next character. Therefore you must use a double backslash (\\) to match a single backslash when not using quotes around the filter value.

The wildcard characters used in the string value of a filter have the following rules:

If an empty string is specified as part of a Name filter, then everything will be matched. Otherwise, an exact match on empty string or default value for the filter is performed if an empty string is specified.

If no wildcards, then an exact match on the filter string is performed.

Wildcard * can be placed anywhere in the filter string and matches zero or more characters.

Wildcard ? can be placed anywhere in the filter string and matches exactly one character.

One character in a set of characters are matched by placing them within [ ]. For example, a[bc] would match 'ab' or 'ac', but it would not match 'ad' or 'abd'.

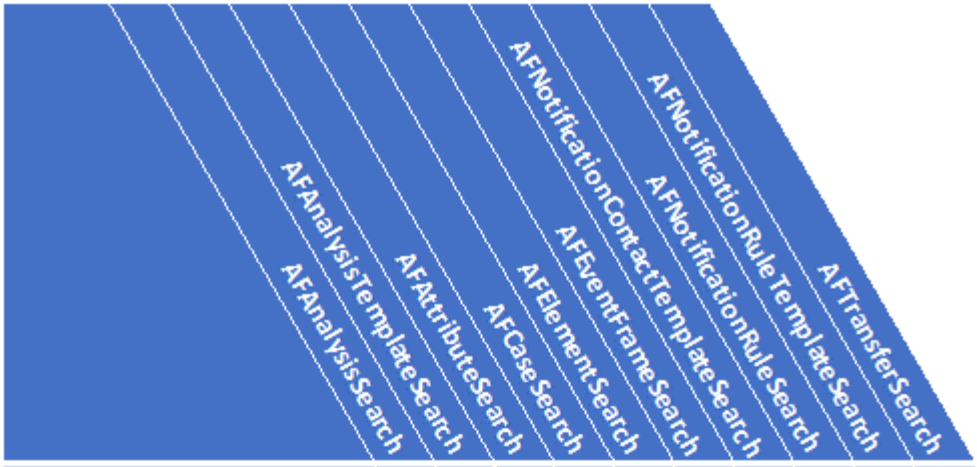
One character in a set of characters are not matched by placing them within [!]. For example, a[!bc] would match 'ad', but it would not match 'ab', 'ac', or 'abd'.

A character in a range of characters from first to last are matched using the following syntax: [first - last]. For example, a[a-c] would match 'aa', 'ab', or 'ac', but it would not match 'ad' or 'abc'.

Filters

Each filter condition is separated by whitespace and will be ANDed together to perform an AFSearch based search query that can be used to search for objects from the AF Server. For more information about the query filters, see the AFSearchFilter topic.

The following chart shows which filters are supported by each search class.



	AFAnalysisSearch	AFAnalysisTemplateSearch	AFAttributeSearch	AFCaseSearch	AFElementSearch	AFEventFrameSearch	AFNotificationContactTemplateSearch	AFNotificationRuleSearch	AFNotificationRuleTemplateSearch	AFTransferSearch
AllDescendants					●	●				
Analysis				◆		●				
AnalysisName						◆				
CanBeAcknowledged						●				
Category	◆	◆	★	◆	●	●		●	●	◆
CategoryName	◆	◆	★	◆	●	●		●	●	◆
Contact							★	✪	✪	
ContactName							★	✪	✪	
CreationDate	✪	✪		✪	✪	✪	★	✪	✪	✪
Description	◆	◆	★	◆	●	●	★	◆	◆	◆
Destination										◆
Duration						●				
Element			★			●				
ElementName						●				
ElementReferenceTemplate						●				
End				◆		●				◆
EventFrame			★							
GroupID	◆	◆								
ID	■	■	■	■	■	■	■	■	■	■
InProgress				◆		●				◆
IsAcknowledged						●				
IsAnnotated				◆	●	●				◆
IsInternal							★			
ModifyDate	✪	✪		✪	✪	✪	★	✪	✪	✪
Name	◆	◆		◆	●	●	★	●	●	◆
OwnerName	❑	❑		❑	❑	❑	❑	❑	❑	❑
Parent					★	★				
PlugIn	★	★	★				★			
PlugInName	★	★	★				★			
ReferenceType					●					
Root					●	●	★			
Sandbox	◆	◆		◆	●	●		●	●	◆
Severity						●				
SortField	◆	◆	★	◆	●	●	★	●	●	◆
SortOrder	◆	◆	★	◆	●	●	★	●	●	◆
Source										◆

The Sandbox filter is not supported by the AFEventFrameSearch if specified with any of the following filters: Analysis, Element, ElementName, or ElementReferenceTemplate. For an AFElementSearch, the Sandbox filter is not supported if the AllDescendants filter is also enabled. The following table describes the dependencies between search filters.

Filter	Notes	Example
AllDescendants	If this filter is enabled and the Root filter is also specified, then the search will be slower because more than one level of the hierarchy must be checked. This filter is enabled by default, so it is recommended to disable this filter if the Root filter is also specified and you do not need to search the full hierarchy under the specified root object.	Root:Tank101 ReferenceType:TankReference
ReferenceType	Finds objects with the specified reference type between the Root object and the found object. If the Root filter is not specified, then finds objects with the specified reference type between the found object and any parent object including the AFDatabase.	Root:Tank101 ReferenceType:TankReference
Value	When using versions of PI AF Server below PI AF Server 2018 (2.10.0.8268), a value filter requires that a Template filter must first be specified that can be used to find the AFAttributeTemplate specified in the value filter.	Template:MyTemplate#1 |Attr#1| Child#2:<=50

Operators

Search operators specify how the filter value is to be compared with the object's value for the filter. For more information about the search operators, see the AFSearchOperator topic.

The following table lists the operators used in the filter condition.

Operator	Description	Example
=	The Equal operator.	Name:Tank* or Name:=Tank*
<>	The NotEqual operator.	"|Attr#1":<>50
<	The LessThan operator.	"|Attr#1":<50
<=	The LessThanOrEqual operator.	"|Attr#1":<=50
>	The GreaterThan operator.	"|Attr#1":>50

Operator	Description	Example
>=	The GreaterThanOrEqual operator.	"|Attr#1":>=50
IN()	The In operator.	"|Attr#1":IN(50;100;150) or Element:IN('Meter'; 'Tank')

Attribute Value Query

On PI AF Server versions below PI AF Server 2018 (2.10.0.8268), an attribute value query requires that a Template filter must first be specified that can be used to find the AFAttributeTemplate specified in the value filter.

When using attribute value queries with a Template filter, search performance can be improved by marking the AFAttributeTemplate as being indexed by setting the IsIndexed property. The Type property of the attribute template determines the type of value to be queried, therefore the type must be defined and not set to a type of object. Any AFAttributeTemplate with a DefaultUOM specified must be of type Single or Double. If a DefaultUOM is not specified in the template for one of these types, then the search will be performed using the specified attribute value as being in the canonical units-of-measure which could cause unexpected results to be returned. The following table shows which AFSearchOperator is supported for each type of value being queried:

Type	Equal	NotEqual	LessThan	LessThanOrEqual	GreaterThan	GreaterThanOrEqual	In
Boolean	Yes	Yes	No	No	No	No	Yes
UInt64	Yes	Yes	No	No	No	No	Yes
String	Yes	Yes	No	No	No	No	Yes
Guid	Yes	Yes	No	No	No	No	Yes
Single	Yes	Yes	Yes	Yes	Yes	Yes	No
Double	Yes	Yes	Yes	Yes	Yes	Yes	No
All Others	Yes	Yes	Yes	Yes	Yes	Yes	Yes

When the Type property of the AFAttributeTemplate is String, then the following restrictions are placed on the string length of the attribute value used in the query:

IsIndexed	Operator	Length Limit
True	Equal or NotEqual	40 characters
True	In	40 characters
False	Equal or NotEqual	No limit
False	In	4096 characters when using PI AF Server 2.6 or greater; otherwise 40 characters

Query Syntax Examples

Below are some example queries:

Get all Event Frames with the name that begins with Tank.

```
query=Tank*

query=name:Tank*

query=Name:=Tank*
```

Get all Event Frames with the name that do not begin with Tank and have the Template name of MyTemplate#1. Note URL encoding.

```
query=Name:<>Tank* Template:MyTemplate%231
```

Get all Event Frames with the name that do not begin with Tank and no Template.

```
query=Name:<>Tank* Template:""
```

Get all Event Frames with the name that do not begin with Tank and have a start time less than 1 week ago and have an end time that is greater or equal to now.

```
query=Name:<>Tank* End:>=* Start:<*-1w
```

Get all Event Frames with the name that have Tank in the name and have a start time greater than or equal to three days ago and that are in progress.

```
query=Name:="*Tank*" Start:>*-3Days InProgress:True
```

Search Modes

The various Search Modes can be recreated with the search query syntax as follows:

StartInclusive

```
query=Start:>=*-1w Start:<*
```

EndInclusive

```
query=End:>*-1w End:<=*
```

Inclusive

```
query=Start:>=*-1w End:<=*
```

Overlapped

```
query=Start:<* End:>*-1w
```

InProgress

```
query=Start:>=*-1w Start:<* InProgress:true
```

The following search modes apply only to event frame searches:

BackwardFromStartTime

```
query=Start:<*-1w SortField:StartTime SortOrder:DESC
```

ForwardFromStartTime

```
query=Start:>=*-1w SortField:StartTime SortOrder:ASC
```

BackwardFromEndTime

```
query=End:<=*-1w SortField:EndTime SortOrder:DESC
```

ForwardFromEndTime

```
query=End:>=*-1w SortField:EndTime SortOrder:ASC
```

BackwardInProgress

```
query=Start:<*-1w InProgress:true SortField:StartTime SortOrder:DESC
```

ForwardInProgress

```
query=Start:>=*-1w InProgress:true SortField:StartTime SortOrder:ASC
```

For More Information

For more information, see the AF SDK Tech Support article on Search Query Syntax

Security Item (Core Services)

Items that can be specified in **GetSecurity**, **GetSecurityEntries** and **GetSecurityEntryByName** actions for asset servers and asset databases. For other resources, only 'Default' security item will be returned.

- Default
  The default security item.
- Analysis
  The Analysis security item.
- AnalysisTemplate
  The Analysis Template security item.
- Category
  The Category security item.
- Contact
  The Contact security item. This is only valid for asset servers.
- Database
  The Asset Database security item. This is only valid for asset servers.
- Element
  The Element security item.
- ElementTemplate
  The Element Template security item.
- EnumerationSet
  The Enumeration Set security item.
- EventFrame
  The Event Frame security item.
- Notification
  The Notification security item.
- NotificationContactTemplate
  The Notification Contact Template security item. This is only valid for asset servers.
- NotificationRule

The Notification Rule security item.

- NotificationRuleTemplate
  The Notification Rule Template security item.
- SecurityIdentity
  The Security Identity security item. This is only valid for asset servers.
- SecurityMapping
  The Security Mapping security item. This is only valid for asset servers.
- ReferenceType
  The Reference Type security item.
- Table
  The Table security item.
- TableConnection
  The Table Connection security item.
- Transfer
  The Transfer security item.

Security Rights (Core Services)

Defines the security access rights of objects.

- None
  There are no security rights granted to the associated object.
- All
  All security rights have been granted to the associated object.
- Read
  There is read access to the associated object.
- Write
  There is write access to the associated object.
- Delete
  There is delete permission to the associated object.
- Execute
  There is execute permission on the associated object.
- Admin
  There is administration permission on the associated object.
- ReadData
  There is read data access to the associated object.
- WriteData
  There is write data access to the associated object.

- Subscribe

  There is subscribe permission on the associated object.
- SubscribeOthers

  There is subscribe other permission on the associated object.
- Annotate

  There is annotate permission on the associated object.
- ReadWrite

  There is read and write access to the associated object.
- ReadWriteData

  There is read data and write data access to the associated object.

Selected Fields (Core Services)

The **selectedFields** parameter filters PI Web API response so that the response only includes a selected set of fields. It allows PI Web API to return none but the information that you are interested in, and therefore reduce your bandwidth usage.

The following examples explain the syntax for the selected fields. Supposedly, we have this JSON response with all fields returned.

```
{
  "Links": {
    "Link1": "https://localhost/link1",
    "Link2": "https://localhost/link2"
  },
  "Items": [
    {
      "WebId": "I1AbEDqD5loBNH0erqeqJodtALA9iLxz4m_5RGAxgAVXYUACw_yrS377vX
0uZVHTgkanusw",
      "Name": "Water",
      "Links": {
        "Self": "https://localhost/piwebapi/attributes/I1AbEDqD5loBNH0erqe
qJodtALA9iLxz4m_5RGAxgAVXYUACw_yrS377vX0uZVHTgkanusw"
      },
      "Value": {
        "Timestamp": "2016-02-22T13:00:50Z",
        "Value": 230.3,
        "UnitsAbbreviation": "m3",
        "Good": "true",
        "Questionable": "false",
        "Substituted": "false"
      }
    },
    {
      "WebId": "I1AbEDqD5loBNH0erqeqJodtALAtcdBko_F5xGTpFCaTFd2TwRQESrwf1o
0iQUMxYbzNaDg",
```

```
      "Name": "Electricity",
      "Links": {
        "Self": "https://localhost/piwebapi/attributes/I1AbEDqD5loBNH0erqe
qJodtALAtcdBko_F5xGTpFCaTFd2TwRQESrwf1o0iQUMxYbzNaDg"
      },
      "Value": {
        "Timestamp": "2016-02-22T13:01:02Z",
        "Value": 56.4,
        "UnitsAbbreviation": "kW",
        "Good": "true",
        "Questionable": "false",
        "Substituted": "false"
      }
    }
  ]
}
```


Response when **selectedFields=Links** is applied:


```
{
  "Links": {
    "Link1": "https://localhost/link1",
    "Link2": "https://localhost/link2"
  }
}
```


Response when **selectedFields=Links.Link1;Links.LinkTwo** is applied:


```
{
  "Links": {
    "Link1": "https://localhost/link1"
  }
}
```


Response when **selectedFields=Items.WebId;Items.Value.Timestamp;Items.Value.Value** is applied:


```
{
  "Items": [
    {
      "WebId": "I1AbEDqD5loBNH0erqeqJodtALA9iLxz4m_5RGAxgAVXYUACw_yrS377vX
0uZVHTgkanusw",
      "Value": {
        "Timestamp": "2016-02-22T13:00:50Z",
        "Value": 230.3,
      }
    },
    {
      "WebId": "I1AbEDqD5loBNH0erqeqJodtALA9iLxz4m_5RGAxgAVXYUACwAujaqyu-e
UKJANgL5jOe0A",
      "Value": {
        "Timestamp": "2016-02-22T13:01:02Z",
        "Value": 56.4,
```

```
            }
        }
    ]
}
```

Severity (Core Services)

The severity of an event frame can be used to designate the importance of the event. The following values are accepted:

- None

  This is the value of an uninitialized severity.
- Information

  The severity level of the event is Information.
- Warning

  The severity level of the event is Warning.
- Minor

  The severity level of the event is Minor.
- Major

  The severity level of the event is Major.
- Critical

  The severity level of the event is Critical.

Sort Field (Core Services)

Indicates which field the items in a collection should be sorted by. The following values are accepted:

- ID

  The returned items are sorted lexicographically by their "Id" field.
- Name

  The returned items are sorted lexicographically by their "Name" field.
- Type

  The returned items are sorted lexicographically by their "Type" field.
- StartTime

  The returned items are sorted chronologically by their "StartTime" field.
- EndTime

  The returned items are sorted chronologically by their "EndTime" field.

The order in which the items are returned may be specified by the Sort Order parameter.

Sort Order (Core Services)

Indicates what order the items in a collection should be returned in. The following values are accepted:

- Ascending

  The returned items are sorted in ascending order (i.e. A to Z).
- Descending

  The returned items are sorted in descending order (i.e. Z to A).

The field by which the items are sorted may be specified by the Sort Field parameter if it is available.

Stream (Core Services)

A stream is defined as a collection of time series data. Currently, only the following resources represent time series data and are considered as streams:

- PI point
- Attribute with a data reference

Note that an attribute without a data reference is not considered as a stream.

Stream Set

A stream set is defined as a collection of streams. Streams inside a stream set can be independent, or share the same base element (e.g. element or event frame) or parent (e.g. parent attribute).

Stream Updates (Core Services)

Stream Updates is a way in PI Web API to stream incremental and most recent data updates for PIPoints/ Attributes on streams and streamsets without opening a websocket. It uses markers to mark the specific event in a stream where the client got the last updates and uses those to get the updates since that point in the stream.

Markers

The marker is a way to mark where in the particular stream the client last got an update. It is used to find any new updates and detect if any updates have been missed by a client. Latest markers are returned both when the client registers for updates and retrieves the updates.

Below are some examples of StreamUpdates URLs:

- POST https://myserver/piwebapi/streams/{webId}/updates - Registering for updates
- GET https://myserver/piwebapi/streams/updates/{marker} - Retrieving updates using a marker
- POST https://myserver/piwebapi/streamsets/updates?{webIds} - Registering for updates for multiple streams
- GET https://myserver/piwebapi/streamsets/updates?{markers} - Retrieving updates from multiple streams

Usage

To use StreamUpdates, the client has to register an attribute or a point for updates by sending a POST request. If registration is successful, then the client can get updates by using the marker in the registration response and sending out the receive updates GET request. There is also an "Updates" link in the response and a link to the latest marker in the "Location" header of the response which can be used directly to get updates.

The response to the GET updates request will include a LatestMarker. This latest marker will now be the current position in the stream and the user can get new updates after this position by sending out GET request using this new marker. These requests can be chained to get incremental updates for registered resources. Stream Updates work in the same way for multiple resources using the streamsets endpoint.

Errors in Retrieval and Registration

When registering for updates, there are 3 possible statuses: 1. Succeded 2. AlreadyRegistered 3. Failed

In case of a Failed Status, clients should inspect the Exception property which will contain the registration errors. If there is an error retrieving updates for a stream, that stream will be unregistered for updates. In the event of an error during retrieval, the client should register for updates again and then request new data to replace the old data (for instance, by calling the `streams/{webId}/recorded` endpoint).

Metadata Changes

Stream Updates does not return any specific metadata/system-level changes to registered attributes or points; however, Stream Updates can notify you when any previously recieved data is no longer valid due to metadata/system-level changes. Some cases where this is possible include: - Changing the data reference for an attribute - Changing the unit of measure for an attribute - Deleting an attribute altogether

In these cases, Stream Updates will return an error stating that the previously returned data is no longer valid. The client can handle these errors like any other by registering for updates again and then requesting new data.

For AF Attributes, an error will be returned if *any* metadata changes are made to the attribute. Note that this does not include static value changes; static value changes will still be returned as normal updates.

For PI Points/Tags, __metadata changes will not return any errors__, unlike AF Attributes. To work around this limitation, it is recommended to use an AF Attribute with a PI Point data reference instead. Another option is to periodically poll for the point's metadata and track any changes clientside.

Other Remarks

- Clients should always use the most up-to-date marker to retrieve updates; previous markers may expire as events get cleaned up periodically.
- Using a Load Balancer with Stream Updates is only supported when the Load Balancer is configured with server affinity.

Sample Client

Below is a sample JavaScript client for StreamUpdates usage :

```
this.latestMarker = null;
function DataQuery() {
    let piWebApi = myServer;
    if (attribute != null) {

    //attribute is the selected attribute by the client
    var webId = attribute.webId;
    var xhr = new XMLHttpRequest();
    var url = "${piWebApi}/streams/updates";

    xhr.open("POST", url, true);

    xhr.onload = function() {
        //Set latest marker
        this.latestMarker = xhr.responseText.LatestMarker;
        }
    xhr.send(webId);
}

// Start up a thread to retrieve updates every 5 seconds
openedThread = setInterval(this.ReceiveUpdates(), seconds(5));
}

function ReceiveUpdates() {
    if (this.latestMarker == null) {
        return;
    }

    let piWebApi = myServer;
    let marker = this.LatestMarker;
    var webId = attribute.webId;
    var xhr = new XMLHttpRequest();
    var url = "${piWebApi}/streams/updates";

    xhr.open("GET", url, true);

    xhr.onload = function() {
        //Update Markers
        this.latestMarker = xhr.responseText.LatestMarker;
        //Display latest Events
        console.log(xhr.responseText.Events)
    }
    xhr.send(this.latestMarker);
}
```

Summary Duration (Core Services)

If specified in hours, minutes, seconds, or milliseconds, the summary durations will be evenly spaced UTC time intervals. Longer interval types are interpreted using wall clock rules and are time zone dependent. For example, an interval created with the string "24h" means using an evenly spaced 24 UTC hour interval between each event. On the other hand, an interval created with the string "1d" would return an interval shorter or longer than 24 hours if the interval encompasses a Daylight Savings Time change. When a positive duration is specified, the summary calculation begins at the earliest bounding time in the timeRange and applies the duration repeatedly in time ascending direction to generate the summary intervals. If a negative duration is specified, the summary calculation begins at the latest bounding time in the timeRange and applies the duration repeatedly in time descending direction to generate the summary intervals.

Note that the order of values returned will still be reflected by the timeRange, regardless of the summary duration sign.

Summary Type (Core Services)

Values to indicate which summary type calculation(s) should be performed. The following values are accepted:

- Total

  A totalization over the time range.
- Average

  The average value over the time range.
- Minimum

  The minimum value over the time range.
- Maximum

  The maximum value over the time range.
- Range

  The range value over the time range (minimum-maximum).
- StdDev

  The standard deviation over the time range.
- PopulationStdDev

  The population standard deviation over the time range.
- Count

  The sum of event count over the time range when calculation basis is event weighted. The sum of event time duration over the time range when calculation basis is time weighted.
- PercentGood

  Percent of data with good value during the calculation period. For time weighted calculations, the percentage is based on time. For event weighted calculations, the percent is based on event count.
- TotalWithUOM

  A totalization over the time range with UOM assigned to the result if the input has units of measure defined. If the input does not have units of measure defined, this returns the same value as Total. If no valid units can be determined, this value will contain an error.

- All

  A convenience for requesting all available summary calculations.
- AllForNonNumeric

  A convenience for requesting all available summary calculations for non-numeric data.

Supported Attribute Data Types (Core Services)

The supported attribute data types are defined below. The following values are accepted

- Anything

  Underlying data type can be any supported type
- Boolean
- BooleanArray
- Byte
- ByteArray
- DateTime
- DateTimeArray
- Double
- DoubleArray
- EnumerationValue

  Values are drawn from an enumeration set. Attributes of this type must specify the name of the enumeration set in the 'TypeQualifier' property.
- Guid
- GuidArray
- Int16
- Int16Array
- Int32
- Int32Array
- Int64
- Int64Array
- Single
- SingleArray
- String
- StringArray

Sync Time (Core Services)

A sync time is an absolute date and time from which to calculate an even set of intervals in a periodic operation. This is useful to establish precise data points, which would otherwise move with every update if the start and end times are relative times. For example, with a start time of "*-12h", an end time of* "", and an interval of "1h", a query would evaluate the current time and return 12 values over the last 12 hours. When repeated, the same query would return a different current time and the pattern of timestamps would not match those returned from the first query. However, when a sync time is specified, the same pattern of timestamps would be repeated.

Boundary Types

Sync time supports Inside and Outside boundary types. Boundary types define the behavior of data retrieval at the end points of a specified time range.

- Inside: Returns the recorded values on the inside of the requested time range as the first and last values.
- Outside: Returns the recorded values on the outside of the requested time range as the first and last values.

PI Web API returns at most two additional values: one before the specified start time, and one after the specified end time. These values are only returned if the sync time shifts the interval such that it no longer aligns with the supplied start or end time.

Time Zones

Sync time, start time, end time, and action can all include time zone information. PI Web API resolves multiple specified time zones in the following order:

1. If the sync time string representation includes a time zone, it is used.
2. If the action specifies a time zone (such as through the `timezone` query parameter), it is used.
3. If no time zone is specified, the PI Web API server time zone is used as a fallback. Note that this fallback is especially important for time zones that are offset by non-hour intervals (for example, India Standard Time, which is UTC+05:30). If a sync time is specified without an explicit time zone, the periodic operation is translated by this additional offset.

Sync Time Examples

In PI Web API, a periodic operation entails taking some action (retrieving data, performing a computation, and so on) at evenly-spaced intervals. To demonstrate these concepts, look at the Stream controller's Interpolated endpoint using the following values:

- A start time of "2014-01-01T01:00:00Z"
- An end time of "2014-01-01T22:00:00Z"
- An interval of "1h"

No sync time

When no sync time is specified, the operation works as usual. This means that the result has the timestamps `{ "2014-01-01T01:00:00Z", "2014-01-01T02:00:00Z", ..., "2014-01-01T22:00:00Z" }`.

Sync time is "1985-08-21T00:30:00Z", and boundary type is "Inside"

With this sync time and boundary type, the result has the timestamps `{ "2014-01-01T01:30:00Z", "2014-01-01T02:30:00Z", ..., "2014-01-01T21:30:00Z" }`. Notice that, even though the

start time was 1:00AM, the first value is at 1:30AM. This is because the "Inside" boundary type retains the results (the range) inside the start and end time (the domain).

Sync time is "1985-08-21T00:30:00Z", and boundary type is "Outside"

With this sync time and boundary type, the result has the timestamps `{ "2014-01-01T00:30:00Z", "2014-01-01T01:30:00Z", ..., "2014-01-01T21:30:00Z", "2014-01-01T22:30:00Z" }`. Notice that the first and last values are outside the start and end time (the domain).

Time Strings (Core Services)

A time string can contain a date, time, time zone information, and a relative interval. The string has two parts, each of which is optional: the date time part and the interval part. The date time part can be "Today" (or "T"), "Yesterday" (or "Y"), a weekday name, a month name, "*" (current time), a null reference, an Empty string, a number less than 32 (day of month), a four digit number greater than or equal 1970 (year), or a date and time specification in any format supported by the Microsoft .NET's System.DateTime.TryParse method. The SQL time string format of hh:mm:ss:fff (three fractional digits are required for this format) is also supported for the time portion of the date time part. A null reference or Empty string is equivalent to "*" and represents the current time.

"Today" and "Yesterday" are the beginning of the specified day at 0 hours local time. A weekday name (either full name or abbreviation) specifies the most current occurrence at 0 hours local time. For example if the current time is Tuesday, specifying a weekday of Wednesday would set the time to the beginning of the day six days ago. A month name (either full name or abbreviation) specifies the current day of the month at 0 hours local time. An absolute date and time specification can be enclosed in double or single quotes (" or ') in any format supported by Microsoft .NET's System.DateTime.TryParse.

When only the time portion of the date time part is specified, then the input is evaluated as the time offset of the current day. If a reference time is specified, then this would be the time offset relative to the day of the reference time.

The interval part follows the date time part in one of the following formats. If the date time part is specified in a DateTime format (format parsed by DateTime.TryParse), then only the first format with interval names specified is allowed unless the date time part is enclosed in double or single quotes (" or ').

```
{+|-}<number>[.<number>] <interval> { [+|-]<number>[.<number>] <interval> }*
```

or

```
{+|-}{ hh | [hh][:[mm][:ss[.ff]]] }
```

Elements in square brackets ([and]) are optional. Alternatives are separated by a vertical bar (|). One selection from the list of alternatives enclosed in braces ({ and }) and separated by vertical bars (|) is required. A plus sign (+) after a group enclosed in braces ({ and }) indicates that zero or more instances of the group is allowed. The following table describes each element.

Element	Description
<number>	A number consisting of one or more digits.
<interval>	The name, short name, or plural name of a standard interval. The table below defines the standard intervals.
+	An optional plus sign, which indicates a positive value.
-	An optional minus sign, which indicates a negative value.
.	A culture-sensitive symbol that separates seconds from fractions of a second. The invariant format uses a period (".") character.
hh	Hours. If hours are omitted, then time separator must be specified before the minutes.
:	A culture-sensitive time separator symbol. The invariant format uses a colon (":") character.
mm	Optional minutes.
ss	Optional seconds.
ff	Optional fractional seconds.

Note to callers

Some formats with missing hours, minutes, and/or seconds that were supported by PI Time are not supported. For example "hh:mm" is supported, but "hh::ss", ":mm:ss", and "::s" are some formats that are not supported. The format must be supported by Microsoft .NET's DateTime.TryParse method.

Some characters used in PI Time strings are not URL-safe. For example, the plus sign ('+') represents a space in a URL. Make sure to URL encode any PI Time strings.

This is a table of the standard intervals. Either the plural full name, singular full name, or short name can be used as the name of the interval. The 'Fractions Allowed' column indicates if a fractional value is allowed for the interval type.

Name	Short Name	Fractions Allowed
millisecond(s)	ms	yes
second(s)	s	yes
minute(s)	m	yes
hour(s)	h	yes
day(s)	d	no
month(s)	mo	no
week(s)	w	no
weekday(s)	wd	no
yearday(s)	yd	no

Examples

- "*" (now)
- "*-8h" (8 hours ago)
- "01" (first of current month)
- "01/01" (first of current year)
- "Monday+8h"
- "Sat, 01 Nov 2008 19:35:00 GMT + 2y+5d-12h+30.55s"
- "Today" (Today at 00:00)
- "T-3d"
- "Yesterday + 03:45:30.25"

Time Type (Core Services)

Defines the timestamp returned for a value when a summary calculation is done. The following values are accepted:

- Auto

  The timestamp returned will be the minimum or maximum time for those summary types, or, the start time of the summary period otherwise.
- EarliestTime

  The timestamp returned will be for the start of the summary period.
- MostRecentTime

  The timestamp returned will be for the end of the summary period.

Time Zone (Core Services)

The time zone can be specified for endpoints with a time (e.g. start time and end time) parameter. The specified time zone context will be used to interpret the time string when possible (e.g. for relative PI time strings such as "y" and "t"). The following behavior is to be expected:

- If the time string already has a time zone offset designated in ISO 8601 format (e.g. "2016-07-07T05:00:00Z" or "2016-07-07T05:00:55+02:00"), the time zone parameter is ignored unless the time range crosses a daylight savings time boundary and an interval is specified. Example detailed below.
  - (streams/interpolated) start time=2020-03-07T12:00:00-02:00, end time=2020-03-09T12:00:00-02:00, interval=1d, timezone=UTC
  - If a daylight savings time boundary falls within the time zone offset start time or end time and there is an interval specified, then the time zone parameter will be honored.
  - In this example, because the two timestamps contain the daylight savings time boundary(March 8th), the returned interpolated timestamps will utilize the time zone UTC specified

Timezone's effect on daylight savings time ranges with intervals

To setup this example the Stream controller's Interpolated endpoint will use the following values:

- A start time of "2020-03-07T12:00:00-02:00"
- An end time of "2020-03-09T12:00:00-02:00"
- An interval of "1d"

No timezone

When no time zone is specified, the timestamps are shifted by one hour after the daylight savings time boundary is passed to account for the time shift. The result has the timestamps `{ "2020-03-07T14:00:00Z", "2020-03-08T13:00:00Z", "2020-03-09T13:00:00Z" }`. The crossover happens on March 8th so we see here that the times are shifted subsequently.

Timezone is UTC

When time zone is specified, the timestamps will be `{ "2020-03-07T14:00:00Z", "2020-03-08T14:00:00Z", "2020-03-09T14:00:00Z" }`. As seen, the time zone parameter will apply to each timestamp in the interval and will each honor the UTC parameter.

Notes about Daylight Savings Setting

- If daylight saving time is enabled on the PI Web API server machine (when applicable):
  - The operating system on which PI Web API instance is installed will dictate the daylight saving time rule for all time zones.
- If daylight saving time is disabled on the PI Web API server machine (when applicable):
  - If no time zone information is specified, the local time on the PI Web API instance is used and no daylight saving time is observed.
  - If the time zone parameter is specified, daylight saving time is observed for all regions that support daylight saving time. This is true even if the time zone specified is identical to the local time zone for the PI Web API server.

Accepted Time Zone Formats

- Windows Time Zone IDs. For example,
  - Eastern Standard Time
  - Pacific Standard Time

  The full list of supported Windows time zone ids on this PI Web API instance can be found here.
- IANA Time Zone Database (or Olson Database) IDs. For example,
  - America/New_York
  - Europe/Paris

  The full list of supported IANA time zone ids on this PI Web API instance can be found here.

Update Option (Core Services)

Indicates how to treat duplicate values, when supported by the data reference. Client applications are responsible for understanding how these options interact with their attributes' data references.

- Replace

  Add the value. If values exist at the specified time, one of them will be overwritten.
- Insert

  Add the value, preserving any values that exist at the specified time.
- NoReplace

  Add the value only if no values exist at the specified time.
- ReplaceOnly

  If a value exists at the specified time, overwrite it. If one does not exist, ignore the provided value.
- InsertNoCompression

  Insert the value without compression.
- Remove

  Remove the value if one exists at the specified time.

User Identity (Core Services)

The user identity can be specified using one of the following three formats:

- User Principal Name

  For example, 'user@domain' or 'user@domain.com'
- Down-Level Logon Name

  For example, 'domain\user'
- User Account Name

  For example, 'user'. The domain of the specified user identity is assumed to be the local DNS domain.

Bearer Authentication

This topic explains using Bearer authentication when building PI Web API client applications. OpenID Connect is a protocol for authentication and is supported by various identity providers. When configured for Bearer authentication, access tokens may be supplied in the Authorization header of a request that provides claims based on the identity configured with the provider.

For information about enabling and configuring Bearer Authentication, please see the Bearer Authentication Settings and Configuration documentation.

Create client credentials

In most cases, it is recommended to first create a new client registration on the identity provider for your application. The client credentials may then be used to request access tokens to be included in PI Web API

requests. The client will need to have the `client_credentials` grant type enabled on the identity provider and `clientid` mappings must be created on any AF or DA resources that the application will be accessing. Read more about client authentication with OpenID Connect here.

While client credentials are the preferred grant type in most cases, others may be used. Read more about OAuth grant types here: https://oauth.net/2/grant-types/

Retrieve an access token

Access tokens can be requested by making a POST request to the token endpoint of the configured identity provider. View the PI Web API OpenId configuration by going to the following endpoint:

`https://{piwebapifqdn}/piwebapi/.well-known/openid-configuration`

The `token_endpoint` property contains the identity provider access token URI. An access token may then be requested by posting to the endpoint with the application `client_id` and `client_secret`, a `grant_type` of `client_credentials` and `scope` should be `system`. See the below curl example:

```
curl --location --request POST 'https://myaimserver.mycompany.com/identity
manager/connect/token' `
--header 'Content-Type: application/x-www-form-urlencoded' `
--data-urlencode 'grant_type=client_credentials' `
--data-urlencode 'client_id=xxxxxxxxxxxx' `
--data-urlencode 'client_secret=xxxxxxxxxxxx' `
--data-urlencode 'scope=system'
```

The resulting response will include an `access_token` that may be used in subsequent requests to PI Web API.

Make a PI Web API request with Bearer authentication

To use Bearer authentication in a PI Web API request, include the `Authorization` header in your request with a value of `Bearer` followed by the access token. For example,

```
curl --location --request GET 'https://{piwebapifqdn}/piwebapi' `
--header 'Authorization: Bearer xxxxxxxxxxx'
```

Cross-Origin Resource Sharing

This topic explains CORS configuration in PI Web API, as well as instructions for setting up CORS correctly.

Background

Cross-Origin Resource Sharing (CORS) is a World Wide Web Consortium (W3C) specification for secure access to resources hosted in a remote domain. This was introduced to overcome the same-origin policy restriction imposed by most modern web browsers. The same-origin policy limits scripts running in one domain from accessing resources that are hosted in another for security purposes. For example, your application that runs on http://my-internal-site.internal/mygreatapp will not be allowed to make AJAX calls to PI Web API running at https://my-pisystem.internal/piwebapi since https://my-internal-site.internal and https://my-pisystem.internal have different origins.

If you are writing an application against PI Web API with scripts running on the browser (e.g., making an AJAX call in your JavaScript application) and seeing error messages in the browser console related to cross-origin requests, you need to enable CORS in the PI Web API configuration. Otherwise, the browser will prevent your script from accessing resources provided by PI Web API.

With CORS set up, the browser sends the request to the PI Web API server and determines whether the request is allowed based on CORS settings in PI Web API. There are two different ways that the browser can ask for permissions. For simplicity:

- Simple cross-domain request: one HTTP request
- Complex cross-domain request: two HTTP requests (preflight request + actual request). The preflight request is made automatically by the browser and uses the OPTIONS method. The actual request is only sent if the permission check is successful for the preflight request.

For more information about simple vs. complex requests, and about CORS in general, consult the following resources:

- W3C recommendation of CORS
- Firefox implementation of CORS
- CORS Support in ASP.NET Web API 2
- The PI Web API User's Guide also contains a section on CORS.

Configuration Settings

By default, CORS is disabled in PI Web API. To enable CORS or change CORS settings, choose one of the following actions:

- Change the CORS related attributes in the AF configuration database under the element OSIsoft > PI Web API > (Your PI Web API Instance Name) > System Configuration.
- Use the PUT action in the Configuration controller.

The following CORS configuration items are available in PI Web API:

- CorsOrigins (default: null)

  The CorsOrigins attribute contains a comma-separated list of domains from which CORS requests may originate. In particular:

  - To indicate cross-domain requests cannot be accepted from any origin, use an empty string. (Effectively disables CORS.)
  - To indicate cross-domain requests are acceptable from any origins, use the asterisk (*) character for wildcard. (Not recommended.)

  Here is an example of a partial simple cross-domain request. The `Origin` header gives the domain of the site that is making the request.

  ```
  GET http://my-pisystem.internal/piwebapi HTTP/1.1
  Origin: http://my-internal-site.internal
  ```

  To allow this request, the CorsOrigins attribute in your PI Web API configuration needs to be set to the wildcard value *, or contain `http://my-internal-site.internal` in the list. If set correctly, the

server will allow the request and set the `Access-Control-Allow-Origin` header. The value of this header either matches the `Origin` header, or is the wildcard value *. Here is a sample response:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://my-internal-site.internal
```

If CorsOrigins attribute is not set correctly, the browser disallows the request and the response does not include the `Access-Control-Allow-Origin header`.

Tips:

- Check the `Origin` header in the request,and make sure the value matches with the header specified in the CorsOrigins attribute.
- Make sure the origin is in correct URL format (e.g., "my-internal-site" is not a valid URL).
- Do not include a space between the comma-separated list of origins.
- Do not include a forward slash at the end of the origins URL.
- Example configuration: `http://my-internal-site.internal,http://my-internal-site2.internal:8080`

- CorsMethods (default: `GET,OPTIONS`)

  The CorsMethods attribute contains a comma-separated list of HTTP methods (verbs) that are allowed in cross-domain requests. In particular:

  - To indicate cross-domain requests cannot be accepted with any HTTP method, use an empty string. (Effectively disables CORS.)
  - To indicate that cross-domain requests are acceptable using any HTTP method, use the asterisk (*) character for wildcard.

  This is only used in a preflight request. Here is an example of a partial preflight cross-domain request.

  ```
  OPTIONS http://my-pisystem.internal/piwebapi/elements/I1EmDqD5loBNH0erqe
  qJodtALAaqQoQHk26BGgMQAVXYR0Ag HTTP/1.1
  Origin: http://my-internal-site.internal
  Access-Control-Request-Method: PUT
  Access-Control-Request-Headers: authorization,content-type
  ```

  Note that it is an OPTION request (preflight). To allow this request, the CorsMethod attribute in your PI Web API configuration needs to be set to the wildcard value *, or contain `PUT` in the comma-separated list. (CorsHeader will be discussed in the next section.) If set correctly, the server will allow the request and set the `Access-Control-Allow-Method` header. Here is a sample response:

  ```
  HTTP/1.1 200 OK
  Access-Control-Allow-Origin: http://my-internal-site.internal
  Access-Control-Allow-Methods: PUT
  Access-Control-Allow-Headers: authorization,content-type
  ```

  Following the preflight request, you will see the actual PUT request and response from the server, similar to the request and response you saw in the simple CORS request in the CorsOrigins section.

  Tips:

  - Check the `Access-Control-Request-Method` request header, and make sure the value matches with the method specified in the CorsMethods attribute.
  - Method names need to be in all upper case.
  - Do not include a space between the comma-separated list of methods.
  - Example configuration: `GET,OPTIONS,POST`

- CorsHeaders (default: null)

  The CorsHeaders attribute contains a comma-separated list of HTTP header keys that are allowed for cross-domain requests. In particular:

  - To indicate cross-domain requests cannot be accepted with any HTTP headers, use a null or empty string.
  - To indicate that cross-domain requests can include any HTTP headers, use the asterisk (*) character for wildcard.

  Similar to CorsMethods, this is only used in the preflight request. Look at the sample preflight request in the CorsMethods section. To allow this request, the CorsHeader attribute in your PI Web API configuration needs to be set to the wildcard value *, or contain `authorization` and `content-type` in the comma-separated list. If set correctly, the server will allow the request and set the `Access-Control-Allow-Headers` header, as shown in the sample response above.

  Tips:

  - Check the `Access-Control-Request-Headers` value in the request, and make sure it matches with the headers specified in the CorsHeaders attribute.
  - Do not include a space between the comma-separated list of headers.
  - If you set CorsHeaders to anything other than `\*`, you should include at least add `accept`, `content-type`, and `origin`, plus any custom headers that you want to support. This is because browsers are not consistent in how they set `Access-Control-Request-Headers`.
  - Example configuration: `accept,authorization,content-type,origin`
- CorsExposedHeaders (default: `Allow,Content-Encoding,Content-Length,Date,Location`)

  The CorsExposedHeaders attribute contains a comma-separated list of HTTP response headers that browsers are allowed to access in addition to the simple response headers exposed by the browser by default. The response headers that are available by default are:

  - `Cache-Control`
  - `Content-Language`
  - `Content-Type`
  - `Expires`
  - `Last-Modified`
  - `Pragma`

  To make other headers available to the application, include them in the list of CorsExposedHeaders attribute. For example, if you are creating an element using CreateElement and would like to get the url resource for the newly created element from the `Location` header, the `Location` header needs to be explicitly specified in CorsExposedHeaders because it is not exposed by default.

  Tips:

  - Do not confuse CorsExposedHeaders with CorsHeaders. The CorsHeaders attribute is used to specify the HTTP headers that are allowed in the requests, while the CorsExposedHeaders attribute is used to specify the HTTP headers that are exposed in the responses.
  - Do not include a space between the comma-separated list of headers.
  - Example configuration: `Allow,Content-Encoding,Content-Length,Date,Location`
- CorsSupportsCredentials (default: `false`)

  The CorsSupportsCredentials attribute indicates whether the browser can send any user credentials with cross-domain requests.

  - False: authentication information sent from the browser is not allowed in the request.
  - True: authentication information sent from the browser is allowed in the request.

An example response when the CorsSupportsCrendentials is set to true:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://my-internal-site.internal
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Headers: authorization,content-type
Access-Control-Allow-Credentials: true
```

Tips:

- CorsSupportsCredentials cannot be used in conjunction with a wildcard (*) value for CorsOrigins.
- Set this flag to `true` if you are using Basic or Kerberos Authentication.
- Example configuration: `true`

- PreflightMaxAge (default: `-1`)

  PI Web API can indicate to client applications how long to cache the preflight request. Set to -1 to never send an Access-Control-Max-Age header in the preflight response. Set to 0 to instruct the client to never cache the preflight response.

Troubleshooting Tips

If you are troubleshooting CORS errors, the browser developer tools (usually opened by pressing F12) provide many useful ways to diagnose the error. Remember that the same-origin policy is a browser implementation, so different browsers may exhibit slightly different behaviors.

1. Look at the error messages in the console. For example,
   - Chrome: `XMLHttpRequest cannot load https://my-internal-site.internal/piwebapi. Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://my-internal-site.internal' is therefore not allowed access. The response had HTTP status code 400.`
   - Firefox: `Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https:// my-internal-site.internal/piwebapi. (Reason: CORS header 'Access-Control-Allow-Origin' missing).`
   - Internet Explorer: `XMLHttpRequest: Network Error 0x80070005, Access is denied.`
2. The browser usually does not give us details on why the error occurs. Even though the browser specifically says that the CORS header `Access-Control-Allow-Origin` is missing, this does not mean that the CorsOrigins attribute is the culprit. This is because the server returns a generic response without any CORS headers if any of the CORS check fails.
3. To identify the misconfigured CORS setting, inspect the request headers.
   - Make sure the value in the `Origin` header in the request matches with one of the origins specified in the CorsOrigins attribute (unless a wildcard is used).
   - Make sure the value in the `Access-Control-Request-Method` in the preflight request matches with one of the methods specified in the CorsMethods attribute (unless a wildcard is used).
   - Make sure the values in the `Access-Control-Request-Headers` in the preflight request match the headers specified in the CorsHeaders attribute (unless a wildcard is used).

- If your client application is requesting for a specific header (e.g., `Location`), make sure it is included in the CorsExposedHeaders list.
- Make sure CorsSupportsCredentials is `true` if you are using Basic or Kerberos authentication.

4. Changing configuration setting does not require a restart of the PI Web API service since the service will update its configuration every 15 seconds or so. To make sure the new configuration values are accepted, check the admin logs in the event viewer for any errors and to verify the modified configuration values.

Dictionary Parameters

Some PI Web API actions accept query string parameters of type **Dictionary Parameter**, representing a set of Name-Value pairs.

Dictionary Parameter Syntax

The Syntax for submitting Dictionary Parameters depends on the name of the *query string parameter* represented by the Dictionary Parameters:

- urlParmeterName[**MyAttributeName**]=**MyValue**

When multiple Name-Value pairs are expected, the query string parameter can be included multiple times in the same query string (see examples below). Any Dictionary Parameter names or values that include RFC3986 reserved delimiters must be percent-encoded.

Dictionary Parameter Examples

If an endpoint accepts a URL Parameter called **Foo**, a single Dictionary Parameter could be provided with this sample URL query:

- ?Foo[Name1]=Value1

If an endpoint accepts a URL Parameter called **Bar**, multiple Dictionary Parameters could be provided with this sample URL query:

- ?Bar[City]=London&Bar[StartTime]=1-1-2017 00:01:00.123

Error Handling

When handling PI Web API responses in your client application, it is recommended to apply error handling logic in order to verify the integrity of your data. This topic will highlight the different types of errors PI Web API may return, and how a client should interpret them.

Status Code

The best indicator for determining the integrity of the response is the response HTTP Status Code. Any HTTP Status Code in the 200-300 range can be considered a successful PI Web API response. Any status code in the 400-500 range indicates a user or server error occurred. These are generally accompanied by a response body

providing a friendly error message to assist with debugging. For more information on interpreting the Status Code returned by PI Web API, see the 'Status Code' section of our 'Getting Started' guide.

Example Response Body:

```
{
    "Errors": [
        "The version of the AF Server does not support the feature Securit
yIdentity."
    ]
}
```

Web Exception

Introduced in PI Web API 2017 R2, PI Web API now exposes a 'WebException' property on **all** controller responses.

Any PI Web API response containing a 'WebException' property which is non-null indicates that PI Web API encountered an **unhandled** error during the transfer of the response stream. These errors can occur despite PI Web API responding with a successful HTTP status code. Responses will not contain a 'WebException' if no error occurred. When present, the 'WebException' property will be present at the top level of all response objects. A 'WebException' contains a 'StatusCode' field, which indicates the correct error HTTP Status Code the client should interpret the response as returning with.

Example Response Body:

```
{
    "WebException": {
        "Errors": [
            "Error occurred during writing of the output stream."
        ],
        "StatusCode": 500
    }
}
```

Stream and Stream Set Controller Responses: Errors

The Stream and Stream Set controller responses may contain an additional field called 'Errors' in the response. The purpose of the 'Errors' field is to indicate that an error occurred for a particular value while streaming the response. For example, if there are 100 values returned and one of them has an associated error, the remaining 99 values do not need to be discarded. Values will not contain an 'Errors' property if no error occurred. Each object in the list of 'Errors' contains which field name caused an error, as well as the exception message.

Unlike 'WebException', which is a single property found at the top level of the response object, an 'Errors' property may be present on any stream value. If a stream value contains an 'Errors' property then its value will be 'null'. If any stream values in the response contain an 'Errors' field, that does not mean the entire value collection returned is invalid.

Example Response Body:

```
{
    "Timestamp": "2014-07-22T14:00:00Z",
```

```
    "UnitsAbbreviation": null,
    "Good": true,
    "Questionable": false,
    "Substituted": false,
    "Value": null,
    "Errors": [
        {
            "FieldName": "UnitsAbbreviation",
            "Message": [
                "PI Point not found."
            ]
        },
        {
            "FieldName": "Value",
            "Message": [
                "PI Point not found."
            ]
        }
    ]
}
```


Notes: There are a couple of things that the user must be aware of when using the Stream or Stream Set controllers.

- Be careful not to confuse the 'Errors' property from the Stream and Stream Set controllers with the 'Errors' field that appears in the body of a response that has a 400- or 500-level status code.
- The Stream and Stream Set controllers do not always handle errors identically. For example, trying to obtain the value of a missing PI Point that results in a 410 response code on the Stream Controller could result in an 'Errors' field appearing on the Stream Set controller.

Impersonation Level

Defines security impersonation levels. Security impersonation levels govern the degree to which a server process can act on behalf of a client process.

- Anonymous

  The server process cannot obtain identification information about the client, and it cannot impersonate the client.
- Delegation

  The server process can impersonate the client's security context on remote systems.
- Identification

  The server process can obtain information about the client, such as security identifiers and privileges, but it cannot impersonate the client. This is useful for servers that export their own objects, for example, database products that export tables and views. Using the retrieved client-security information, the server can make access-validation decisions without being able to use other services that are using the client's security context.

- Impersonation

  The server process can impersonate the client's security context on its local system. The server cannot impersonate the client on remote systems.

- None

  An impersonation level is not assigned.

Multipart Requests

This topic is only a brief overview of multipart requests; more information is available in the associated IETF RFC2046.

HTTP requests may declare their media type to be "multipart". This is accomplished by setting the `Content-Type` header to a MIME type whose type is `multipart` (for example, `multipart/form-data`). This indicates to the recipient that the body of the request is formatted as a series of "body parts" grouped together as a single "multipart" request. Each part starts with a boundary to indicate to the recipient the start of a new body part. The boundary is specified by setting the `boundary` property in the `Content-Type` header.

Continuing the previous example of `multipart/form-data`, the `Content-Type` header would look like `multipart/form-data; boundary=MYBOUNDARY`. Then, the body parts contained in the request body would appears as follows:

```
--MYBOUNDARY

This is my first body part.
--MYBOUNDARY

This is my second body part.
--MYBOUNDARY--
```

Observe that the last boundary is prefixed by `--` to indicate the start of a new body part, and that the last body part declares the end of the body by suffixing another `--`.

Each "body part" is similar (but not equivalent) to a normal HTTP request. It is formatted as a header area and a body area separated by a blank line. The header area is allowed to contain `Content-` headers, but can be left blank to indicate plain text. The body area is similar to a normal request body, except that the **boundary may not appear within it** - if it does, the recipient won't be able to determine the intended body parts.

Example

The following example is a complete multipart HTTP request to a non-existent URI. Our imaginary URI expects three body parts: a snippet of text, a JSON-formatted object, and a Rich Text Format document. The first line is the HTTP request's Request-Line; this is **not** part of the request body. The next two lines are the `Host` and `Content-Type` headers, and are also **not** part of the request body.

```
POST /an/example/path HTTP/1.1
Host: www.mywebsite.com
Content-Type: multipart/form-data; boundary=MYBOUNDARY

--MYBOUNDARY
```

```
    This is the body area of the first body part. Note that this body part
has an empty header
    area (indicating that the content is plain text), and that a blank lin
e separates the
    header area and body area.
    --MYBOUNDARY
    Content-Type: application/json

    {
        "ExampleMessage": "This is the body area of the second body part."
,
        "Notes": [
            "Note that the header area of this body part contains a Conten
t-Type header.",
            "Note that the header area and the body area are separated by
a blank line."
        ]
    }
    --MYBOUNDARY
    Content-Disposition: form-data; name="anExample"; filename="sample.rtf"
    Content-Type: application/rtf

    {\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1033{\fonttbl{\f0\fnil\
fcharset0 Calibri;}}
    {\*\generator Riched20 10.0.16299}\viewkind4\uc1
    \pard\sa200\sl276\slmult1\f0\fs22\lang9 This is a sample RTF document.
 Note that in this body
    part, we have specified a \i Content-Disposition\i0  header. This heade
r allows for additional
    useful information to be passed to the recipient, such as the filename
of the document. At
    this end of this body part, we mark the end of the multipart request.\
par
    }
    --MYBOUNDARY--
```

Query String

The query string (or match pattern) can include regular characters and wildcard characters. Regular characters must match exactly the characters specified in the query string. Wildcard characters can be matched with arbitrary fragments of the query string. Wildcard characters can be escaped using the single backslash () character. Use a double backslash (\) to match a single backslash. The syntax of the query string has the following rules:

- If null or empty string, then everything will be matched.
- If no wildcards, then an exact match on the query string is performed.
- Wildcard * can be placed anywhere in the query string and matches zero or more characters.

- Wildcard **?** can be placed anywhere in the query string and matches exactly one character.
- One character in a set of characters are matched by placing them within **[ ]** . For example, **a[bc]** would match 'ab' or 'ac', but it would not match 'ad' or 'abd'.
- One character in a set of characters are not matched by placing them within **[!]** . For example, **a[!bc]** would match 'ad', but it would not match 'ab', 'ac', or 'abd'.

Run State

PI Web API server run state.

- Unknown

  The PI Web API state is unknown.
- Running

  PI Web API is running.
- Stopping

  PI Web API is stopping (a stop operation is pending). This typically occurs when stopping the PI Web API service while there are outstanding requests being processed.
- Stopped

  PI Web API is stopped.

URL Encoding

Some common characters used by PI System features, such as AF resource names, may produce errors when included in a request URI. These characters must be percent-encoded according to RFC 3986, which defines the rules for URIs. Characters that must be percent-encoded are referred to as "reserved characters". For example, if you are trying to include an AF Element name like `Element#&StartTime=01-01-17` in your request URI, you would need to percent-encode it like `Element%23%26StartTime%3D01-01-17`.

Table with common url encodings

Symbol	Escape Character
Space	%20
!	%21
"	%22
#	%23
$	%24
%	%25
&	%26
'	%27
(	%28
)	%29

Symbol	Escape Character
*	%2A
+	%2B
,	%2C
-	%2D
.	%2E
/	%2F
:	%3A
;	%3B
<	%3C
=	%3D
>	%3E
?	%3F
@	%40

Value Types

The PI Web API supports values of the following underlying types:

- Boolean
- Byte
- DateTime
- Double
- Guid
- Int16
- Int32
- Int64
- Single
- String
- Arrays of the above types
- Enumeration values

Attributes marked with one of these value types, as well as attributes of type 'Anything' whose values are of one of these types, are supported by the PI Web API. Other data types will be rejected on input. On output, the type field of attributes marked with an unsupported type will display 'Unsupported', and retrieving values of unsupported type will return status code 501.

Good values (values for which the 'Good' property is set to `true`) are mapped to and from JSON according to the following rules:

- Boolean: JSON boolean
- Byte, Double, Int16, Int32, Int64, Single: JSON number
- DateTime: ISO8601 UTC formatted JSON string
- Guid: hyphenated hexadecimal digits as a JSON string
- String: JSON string

- Array: JSON array of the corresponding type
- Enumeration value: JSON object with 3 properties: Name (string), Value (number), IsSystem (boolean)

Bad or error values (values for which the 'Good' property is set to `false`) may be serialized as one of the above types or, more commonly, as an error represented as a string, enumeration value, or JSON object containing a single property, 'Errors', which is a string array containing one or more error messages.

The above discussion applies to default PI System behaviors. Custom data references may produce other behavior.

WebID

Resources in PI Web API are addressed by WebIDs, which are persistent, URL-safe identifiers that encode the GUIDs and/or paths associated with objects in the PI System (see: WebID Encoding). WebIDs are often used in links, and can also be found inside JSON responses. Client applications can then use these WebIDs as opaque identifiers in other URLs or query parameters. Because WebIDs are persistent, clients may cache URLs containing WebIDs for future use, even in newer versions of PI Web API.

WebID version 2.0, introduced in PI Web API 2017 R2, provides different types of WebIDs (see: WebID Type). Specifying the WebID type gives you options for reducing WebID sizes (for URL length limitations), for identifying ambiguous paths/names of AF Event Frames and AF Notifications, and for accommodating path and name changes. Any existing WebIDs (from version 1.0) can still be used with newer versions.

2019 - Detailed Changes (PI Web API OMF Services)

New Concept: OMF Endpoint

PI Web API 2019 introduces the PI Web API OMF Services plugin. This plugin adds an Open Message Format (OMF) endpoint to the PI Web API, which can be used to ingress OMF messages to your PI AF and PI Data Archive servers. For more information about OMF, view the OMF Endpoint Notes topic.

New Concept: CREATE Action

The OMF specification defines the CREATE action. When a CREATE message is received, it will be treated as an create-and-assert operation. The PI Web API will attempt to create PI System resources corresponding to the contents of the message. If an existing resource exists, if the resource does not match what the PI Web API is trying to create, the request will fail. The type of resource created by the PI Web API depends on the type of OMF message being used.

New Concept: DELETE Action

The OMF specification defines the DELETE action. When a DELETE message is received, it will be treated as a try-remove operation. The PI Web API will look for the specified resource and compare it to the contents of the message: if the message matches the existing PI System representation, the resource will be deleted. If the resource is not found, no change will occur.

New Concept: Types

The OMF specification introduces the concept of a *type.* A *type* describes the contract that later OMF messages should be held to. The PI Web API OMF Services plugin will create an AF Element Template to represent a *type.*

New Concept: Containers

The OMF specification introduces the concept of a *container.* A *container* represents a destination that later OMF messages can write data to. The PI Web API OMF Services plugin will create one or more PI Points to represent a *container.*

New Concept: Data

The OMF specification introduces the concept of *data. Data* can be broken down into several types: *static, dynamic,* and *link* data.

Static Data

The PI Web API OMF Services plugin will create an AF Element to represent *static data.*

Dynamic Data

The PI Web API OMF Services plugin will create time-series data, stored in PI Points, to represent *dynamic data.*

Link Data

Link data is a subset of *static data* using the reserved OMF type `__link`. *Link data* represents hierarchical information. The PI Web API OMF Services plugin only support only implements two types of *link data:* static-to-static links, and static-to-dynamic links.

Static-to-static links are links that have a static data source and a static data target. The PI Web API OMF Services plugin will create an AF Element hierarchy to represent static-to-static links.

Static-to-dynamic links are links that have a static data source and a container target. The PI Web API OMF Services plugin will create one or more AF Attributes to represent static-to-dynamic links. The created AF Attributes use the PI Point Data Reference to read the time-series data stored in the PI Points that represent the container.

Schema Changes

For detailed information about the PI Web API OMF Services storage schema, refer to the Developer Companion Guide.

Types

- Introduced

Containers

- Introduced

Data

- Introduced

2019 SP1 - Detailed Changes (PI Web API OMF Services)

New Concept: Simple Types

PI Web API OMF Services 2019 SP1 introduces the *simple type* concept. A *simple type* is a dynamic type that has a single value property. *Simple types* are not part of the OMF specification: they are specific to the PI Web API. When discussing types, the term *complex types* is used to describe dynamic types with more than one value property.

Using a simple type changes how some resources are represented in your PI AF or PI Data Archive:

Comparison Detail	Simple Type	Complex Type

Type message	[{
	"id": "simple-dynamic-type1",
	"version": "1.0.0.0",
	"type": "object",
	"classification": "dynamic",
	"properties": {
	"time": {
	"type": "string",
	"format": "date-time",
	"isindex": true
	},
	"attribute1": {
	"type": "string",
	"name": "Attribute One"
		[{
		"id": " complex-dynamic-type1",
		"version": "1.0.0.0",
		"type": "object",
		"classification": "dynamic",
		"properties": {
		"time": {
		"type": "string",
		"format": "date-time",
		"isindex": true
		},
		"attribute1": {
		"type": "string"
		"name": "Attribute One"

	}
	}
	}]
		},
		"attribute2": {
		"type": "string"
		"name": "Attribute Two"
		}
		}
		}]

Container message	[{ "id": "simple-container1", "name": "Simple Container", "typeid": "simple-dynamic-type1" }]	[{ "id": "complex-container1", "name": "Complex Container", "typeid": "complex-dynamic-type1" }]
PI Points created in version 2019	simple-container1.attribute1	complex-container1.attribute1 complex-container1.attribute2
PI Points created in version 2019 SP1	simple-container1	complex-container1.attribute1 complex-container1.attribute2
AF Attributes created in version 2019 (using __link)	Simple Container.Attribute One	Complex Container.Attribute One Complex Container.Attribute Two
AF Attributes created in version 2019 SP1 (using __link)	Simple Container	Complex Container.Attribute One Complex Container.Attribute Two

New Concept: UPDATE Action

PI Web API 2019 SP1 introduces limited UPDATE action support. UPDATE is supported for DATA messages. UPDATE is **not yet implemented** for TYPE or CONTAINER messages.

When an UPDATE message is received, it will be treated as an insert-or-replace operation. If the specified resource already exists, it will be replaced. If the specified resource does not exist, it will be created.

New Concept: Migrations

PI Web API 2019 SP1 introduces limited OMF migration support. This migration support allows somes resource types created using prior versions of PI Web API OMF Services — as well as non-OMF resources — to be re-purposed for use with the current version of PI Web API OMF Services. When a migration occurs, the resource will be modified for use with the PI Web API — sometimes destructively.

When a `CREATE LINK DATA` message is received, AF Attribute migration may occur. If a link between a static instance and a container already exists, AF Attributes belonging to that link may be renamed: for example, from `Simple Container.Attribute One` to `Simple Container`.

When a `CREATE CONTAINER` message is received, PI Point migration may occur. If a PI Point exists with the name PI Web API was going to use, it will be re-purposed for use with PI Web API OMF Services.

When migrating a PI Point, the way it was created affects how it will be migrated for use with PI Web API OMF Services 2019 SP1. PI Points are grouped in the following manner for migration:

- PI Points created by OMF (PI Web API 2019 and 2019 SP1) contain valid Point Source and Extended Descriptor attributes.
- PI Points that were not created by OMF have an Extended Descriptor attribute that the PI Web API cannot parse.

For PI Points created using OMF (PI Web API 2019), PI Points belonging to containers using a simple type will be renamed: for example, from `simple-container-id.attribute1` to `simple-container-id`.

For any migrated PI Point — regardless of how it was created — some PI Point attributes will be overwritten:

- The `OmfSchemaVersion=1` key-value pair will be added to the PI Point Extended Descriptor attribute.
- The PI Point Point Type will be changed according to the type being used by the container. The Data Archive controls whether the point type conversion is possible.
- The container description will be stored in the PI Point Descriptor attribute. If the container does not have a description, the PI Point Descriptor attribute will be blank. This will overwrite an existing Descriptor attribute value.

Schema Changes

Types

- No changes

Containers

- PI Point naming conventions have changed. Previously, all OMF PI Points were named like `ContainerId.PropertyIndex`. Now, PI Points using simple types will be named like `ContainerId`. The naming convention for PI Points using complex types has not been changed.
- The container description will be stored in the PI Point Descriptor attribute of all PI Points belonging to that container.

- The PI Point Extended Descriptor attribute now includes an additional key-value pair with the key `OmfSchemaVersion`. The value of this pair indicates which version of the PI Web API OMF Services was used to create the PI Point. A value `1` indicates that the PI Point was created using PI Web API OMF Services 2019 SP1. If this key-value pair is not present in a PI Point's Extended Descriptor attribute, it is assumed that the PI Point was created using PI Web API OMF Services 2019 (the first release of the plugin.)

Data

- AF Attribute naming conventions have changed. Previously, creating a static-to-dynamic link resulted in AF Attributes named like `ContainerName.PropertyName`. Now, static-to-dynamic links for a container using a simple type will result in AF Attributes named like `ContainerName`. The naming convention for AF Attributes using complex types has not been changed.

2021 - Detailed Changes (PI Web API OMF Services)

New Concept: OMF version 1.2

PI Web API 2021 introduces partial support for version 1.2 of the OMF specification. A client can indicate that OMF v1.2 should be used by setting the `omfversion` request header to the value `1.2`. OMF v1.2 builds upon OMF v1.1, and introduces many new features. For more information about OMF v1.2, consult the OMF v1.2 Specification.

The following OMF v1.2 features are supported by the PI Web API:

- The TYPE message `enum` property.
  - Although `ENUM` TYPEs are fully supported, there are some PI System limitations. See the `ENUM` TYPE section for more information.
- The TYPE property `isQuality` property.
- The TYPE property `reftypeid` property.
  - Currently, only `ENUM` TYPE reference types can be used.
- The TYPE property `maximum` and `minimum` properties.
- The `CONTAINER` message `dataSource` property.
- The `CONTAINER` message `propertyOverrides` property. The following property overrides are supported:
  - `minimum`
  - `maximum`
  - `uom`
- The `__link` type's `property` property.
  - Currently, the `property` property is only support for static-to-dynamic links.

The following OMF v1.2 features are not yet supported by the PI Web API:

- The TYPE message `extrapolation` property.
- The TYPE message `baseTypeId` property.
- The TYPE property `interpolation` property.

- Non-ENUM TYPE messages that omit the `classification` property.
- The CONTAINER message `extrapolation` property.
- The CONTAINER message `metadata` and `tags` properties.
  - These properties already existed in OMF v1.1, but the PI Web API silently ignored them. The PI Web API will now include a warning indicating support for these properties is not yet implemented.
- The following CONTAINER message `propertyOverrides` properties are not yet supported:
  - `name`
  - `description`
  - `interpolation`
- The DATA message `properties` property.
- Type-to-type LINK messages.

New Concept: ENUM TYPEs

OMF version 1.2 introduces ENUM types. An ENUM type is used to restrict a property's values to a known set of legal values. To specify that a TYPE property uses an ENUM, use the new `reftypeid` property instead of `type` and `format`. When an ENUM type is created, the PI Web API will always create an AF Enumeration Set to represent the ENUM, and may create a DA Digital Set if the ENUM is eligible. An ENUM type is eligible for a DA Digital Set if it meets the following criteria:

- The ENUM must be of format `int16` or `int32`.
- The ENUM must start at zero.
- The ENUM must increment monotonically in steps of one.

An ENUM TYPE AF Enumeration Set is of the following schema:

- The Enumeration Set's name must be the ENUM's ID.
- The Enumeration Set's description must be a set of key-value pairs. Keys are case-sensitive. Keys must be unique. Values containing the characters `,`, `"`, `\`, `=`, or leading/trailing whitespace must be enclosed in double-quotes (`"`). Values enclosed in double-quotes must escape the characters `\` and `"` with a leading backslash (`\`). The following keys are required:
  - `OmfSchemaVersion` - The schema version being used. The value must be `0`.
  - `OmfTypeName` - The ENUM's name.
  - `OmfTypeVersion` - The ENUM's version.
  - `OmfTypeDescription` - The ENUM's description.
  - `OmfEnumType` - The ENUM's type.
  - `OmfEnumFormat` - The ENUM's format.
  - `OmfEnumNullable` - Indicates whether the ENUM is nullable. When the ENUM is nullable, NULLABLE; otherwise, NOTNULLABLE.
- The Enumeration Set's values must be of the following schema:
  - The Enumeration Value's value must be set according to the ENUM's type and format as follows:

Type	Format	Value
`integer`	`int64`	The value is not used by PI Web API.

integer	int32	The value must be the numerical value of the ENUM value.
integer	int16 *(Default)*	The value must be the numerical value of the ENUM value.
integer	uint64	The value is not used by PI Web API.
integer	uint32	The value is not used by PI Web API.
integer	uint16	The value must be the numerical value of the ENUM value.

- The Enumeration Value's name must be the ENUM value's name.
- The Enumeration Value's description must be a set of key-value pairs. The following keys are required:
  - OmfEnumValue - The numerical value of the ENUM value.
    - AF Enumeration Value values must be int32; this would make valid OMF ENUMs using formats like int64 unable to be represented in AF. As a workaround, the PI Web API stores the numerical value in the description.

An ENUM TYPE Digital Set is of the following schema:

- The Digital Set name must be the ENUM's ID.
  - ex. If your ENUM has an ID of My.Example.Enum, the name of the Digital Set must be My.Example.Enum.
- The Digital State values must be of the following schema:
  - The name must be the ENUM value's name. This field is case-insensitive.
    - The Data Archive deduplicates Digital State names case-insensitively. This means once a Digital State named like FOO exists, an attempt to create a new Digital State foo will succeed, but subsequent reads will return the name of the previously-created FOO. As a workaround, the PI Web API treats Digital State names case-insensitively.
  - The numerical value must be the ENUM value's numerical value.

New Concept: Quality Mappings

The OMF version 1.2 specification introduces the isQuality property to TYPE properties; however, the specification does not define how an OMF server implementation utilizes this field. PI Web API 2021 includes a *quality mapping* feature that allows an application to write quality information to the Data Archive. This quality mapping mechanism is not part of the OMF specification - it is a PI Web API specific feature.

A quality mapping can be associated with a DYNAMIC TYPE that has a property marked as isQuality. To associate a quality mapping with a DYNAMIC TYPE, include a metadatum with key DataQualitySchema and value of the quality mapping's ID. When a DYNAMIC DATA value is received, the value associated with the quality property will be compared to the quality mapping. The comparison can be either a direct value mapping (ex. "A value of '5' means 'GOOD', a value of '12' means 'QUESTIONABLE'"), or a set of flags (ex. "A value is 'BAD' if at least one of these bits is set"). The type of comparison is defined at the quality mapping level - it cannot mix value comparisons and flag comparisons.

Creating custom quality mappings is discouraged. As a PI Web API specific feature, quality mapping behavior may change between future product or OMF versions, making maintaining custom quality mappings difficult. For more information on custom quality mappings, please contact your technical support representative.

Quality mappings are stored as an AF Enumeration Set on the OMF AF Asset Database according to the following rules:

- The Enumeration Set name must have the prefix `__qualitymap.`, followed by your quality map's ID. This is needed for the PI Web API to recognize your Enumeration Set as a quality map.
  - ex. If you sent the metadata value `Foo`, your Enumeration Set name should be `__qualitymap.Foo`.
- The Enumeration Set description must be a set of key-value pairs. Keys are case-sensitive. Keys must be unique.
  - The following keys are required:
    - `Type` - The OMF property type that the values of this quality mapping are defined in.
    - `Format` - The OMF property format that the values of this quality mapping are defined in.
    - `IsNullable` - Indicates whether a value of `null` should be treated as `Good`. If `true`, `null` will be treated as `Good`; otherwise, `null` will be treated as having the quality associated with the lowest numerical value in the quality map.
    - `IsFlags` - Indicates whether the values contained by this quality mapping should be treated as literal values, or as masks. If `true`, incoming values will be masked against the values defined by the quality mapping, and the worst quality for which all bits in the flag were set will be returned. An incoming value of zero will be treated as `Good` unless the quality mapping includes an explicit value of 0. If `false`, if an incoming value is contained by the quality map, the associated quality will be used; otherwise, `Good` will be used.
  - The following keys are optional:
    - `Source` - A human-readable indicator describing the source of the quality mapping.
    - `Name` - A human-readable name associated with this quality mapping.
    - `Mask` - When present, incoming values will be masked by this value. This value must be of the `Type` and `Format` used by the quality mapping.
- The Enumeration Values must be of the following form:
  - The value does not matter.
  - The name must be a valid value according to the enumeration's description's `Type` and `Format` keys. Values must be unique. This is the value that incoming `DATA` values will be compared against.
    - The quality mapping value is stored in the Enumeration Value's name, rather than the Enumeration Value's value, because AF Enumeration Set values must be within the range of `int32`. This means some OMF property formats, like `int64`, would not be able to be represented if the Enumeration Value's value was used.
  - The description must be a set of key-value pairs. Keys are case-sensitive. Keys must be unique. The following keys are optional:
    - `Quality` - The quality associated with this value. If not specified, a quality of `Good` is assumed. The value must be one of the following:
      - `Good`
      - `Questionable`
      - `Bad`
    - `SystemStateCode` - A manual system state code to use when storing quality data. This value is only allowed if `Quality` is set to `Bad`. The specified value must be defined by the AFSDK's `AFSystemStateCode` enumeration.

- `Comment` - A human-readable comment associated with this value.

New Feature: Improved Per-User OMF Cache Control

PI Web API uses a per-user cache to improve request performance. Developers that have used PI Web API Core Services may already be familiar with this mechanism: When a user makes a request to the PI Web API, if they have recently used a PI System resource, the cached version will be used. This can greatly improve client application performance and reduces load on the PI System. However, some applications may need to bypass the PI Web API's per-user cache (ex. they cannot tolerate stale data). Applications that need to bypass the PI Web API's per-user cache do so by including the `Cache-Control: no-cache` header on their request. Note that if a client application requests cache refreshes too often, it can severely impact PI System performance. **Clients must use this feature with caution.**

PI Web API 2021 improves application control over the PI Web API per-user cache. In previous versions, the OMF cache could not be cleared by the user; applications either needed to wait for the cache to automatically referesh, or had to restart the PI Web API service. OMF applications can now trigger a refresh of the PI Web API per-user cache using the same `Cache-Control: no-cache` header. Note that this header is **not part of the OMF specification** - the per-user cache is a PI Web API specific feature intended to improve application performance.

Expanded Feature: UPDATE Action

PI Web API 2021 expands support for the `UPDATE` action. Previously, the `UPDATE` action was supported only for `DATA` messages. Now, the `UPDATE` action can be used for both `CONTAINER` and `DATA` messages. `UPDATE` is **not yet implemented** for `TYPE` messages.

The PI Web API's `UPDATE CONTAINER` feature can update any `CONTAINER` field, including `typeId`. When changing an existing `CONTAINER`'s type, the new `TYPE` must not add or remove any value properties. This is a PI Web API specific restriction - it is not part of the OMF specification. This restriction exists so that an existing `CONTAINER` does not change from a simple type to a complex type (or vice versa), which could trigger unintentional PI Point renaming.

Breaking Change: PI Point Schema Version 2

As part of PI Web API 2019 SP1's resource migration support, the concept of schema versions was introduced. Schema versions describe the structure of PI System resources that are created or interacted with by the PI Web API; schema versions are not part of the OMF specification. PI Web API 2021 introduces a new PI Point schema version: schema version 2. Schema version 2 introduces greater control over PI Points created using OMF, and supports new OMF features such as property `minimum` and `maximum`.

PI Points will automatically be upgraded to schema version 2 when a `CREATE CONTAINER` or `UPDATE CONTAINER` message is received. PI Points are upgraded to schema version 2 regardless of the version of OMF being used. OMF client application behavior is not impacted; no changes are required. PI System applications may experience changes in behavior (ex. PI Vision symbol axes scaling changing based on the PI Point Span and Zero); for this reason, this change is considered a breaking change.

Each PI Point maps to a value property on the `TYPE` used by a `CONTAINER`. PI Point attributes are mapped using schema version 2 as follows:

<table>
<tr><th>PI Point Attribute</th><th>OMF Mapping</th></tr>
<tr><td>Name (AKA Tag)</td><td>

If the CONTAINER type is simple (has one value property), of the form <code>$"{container.Id}"</code>.
If the CONTAINER type is complex (has more than one value property), of the form <code>$"{container.Id}.{property.Index}"</code>.

For more details on simple and complex types, see the PI Web API OMF Services 2019 SP1 detailed changes topic.
</td></tr>
<tr><td>Descriptor</td><td>The CONTAINER's <code>description</code>. If the CONTAINER does not have a <code>description</code>, this attribute is not set.</td></tr>
<tr><td>Point Source</td><td>

If the CONTAINER specifies a <code>dataSource</code>, the specified <code>dataSource</code>.
If the CONTAINER does not specify a <code>dataSource</code>, the default value <code>PIWebAPI_OMF</code>.

</td></tr>
<tr><td>Point Type</td><td>

If the property is not using <code>reftypeid</code>, derived from the property's <code>type</code> and <code>format</code> according to the table below.
If the property is using <code>reftypeid</code>, if the referenced ENUM corresponds to a Digital Set, <code>Digital</code>. Otherwise, derived from the ENUM's <code>type</code> and <code>format</code> according to the table below.

<table>
<tr><th>Type</th><th>Format</th><th>PI Point Type</th><th>Notes</th></tr>
<tr><td><code>array</code></td><td></td><td>Unsupported</td><td></td></tr>
<tr><td><code>boolean</code></td><td></td><td><code>Int16</code></td><td></td></tr>
<tr><td><code>integer</code></td><td><code>int64</code></td><td>If this PI Point is using <code>reftypeid</code>, <code>String</code></td><td>The Data Archive does not have a numerical data type large enough to store an</td></tr>
</table>
</td></tr>
</table>

		• If this PI Point is not using `reftypeid`, `Float64`	`int64` without precision loss.
`integer`	`int32` *(Default - non-*`ENUM`*)*	`Int32`	
`integer`	`int16` *(Default -* `ENUM`*)*	`Int32`	The Data Archive has special behavior for `Int16` PI Points. For consistency, the PI Web API instead uses `Int32`.
`integer`	`uint64`	• If this PI Point is using `reftypeid`, `String` • If this PI Point is not using `reftypeid`, `Float64`	The Data Archive does not have a numerical data type large enough to store a `uint64` without precision loss.
`integer`	`uint32`	`Float64`	The Data Archive does not have a

			native unsigned integer data type. However, a `Float64` can store a `uint32` without precision loss.
`integer`	`uint16`	`Int32`	The Data Archive does not have a native unsigned integer data type. However, an `Int32` can store a `uint16` without precision loss.
`number`	`float64`	`Float64`	
`number`	`float32` *(Default)*	`Float32`	
`number`	`float16`	`Float16`	Although the Data Archive supports `Float16`, because the Asset Framework server does not, properties of type `float16` cannot be created.
`object`	`dictionary`	*Unsupported*	

	<table><tr><td>string</td><td></td><td>String</td><td></td></tr><tr><td>string</td><td>date-time</td><td>Timestamp</td><td></td></tr></table>
Digital Set Name	• If the property is using reftypeid and the specified ENUM is eligible for a Digital Set, the Digital Set with the name matching the ENUM's ID. • If the property is using reftypeid and the specified ENUM is not eligible for a Digital Set, this attribute is not set. • If the property is not using reftypeid, this attribute is not set.
Eng Units	In order of precedence, 1. If the CONTAINER contains a property override for this property that specifies a uom, the specified uom. 2. If the property specifies a uom, the property's uom. 3. This attribute is not set.
Zero	In order of precedence, 1. If the CONTAINER contains a property override for this property that specifies a minimum, the specified minimum. 2. If the property specifies a minimum, the property's minimum. 3. The Classic PI Point Class' default zero.
Span	In order of precedence, 1. If the CONTAINER contains a property override for this property that specifies a maximum, a value derived from the specified maximum. 2. If the property specifies a maximum, a value derived from the property's maximum. 3. The Classic PI Point Class' default span.
Step	Set according to the Point Type and property type and format. The Point Type table takes precedence over the property type and format. Unless otherwise noted, the Step will be set to 0.

Point Type	Step Value
String	1
Timestamp	1
Blob	1
Digital	*Unsupported*

Type	Format	Step Value
array		*Unsupported*
boolean		1
integer	int64	• If this PI Point is using reftypeid, 1 • If this PI Point is not using reftypeid, 0
integer	int32 *(Default - non-*ENUM*)*	0
integer	int16 *(Default -* ENUM*)*	0
integer	uint64	• If this PI Point is using reftypeid, 1 • If this PI Point is not using reftypeid, 0
integer	uint32	0
integer	uint16	0
number	float64	0
number	float32 *(Default)*	0
number	float16	0

	`object` | `dictionary` | *Unsupported*
	`string` | | `1`
	`string` | `date-time` | `1`

Extended Descriptor	The extended descriptor is reserved for internal PI Web API use.

Breaking Change: Parameter Names

When the PI Web API service has been configured for debug mode, PI Web API OMF Services may include additional information when an event is returned as part of the response payload. These pieces of additional information are called *parameters*. These parameters are meant to be used by developers when debugging OMF client applications, and are not intended for programmatic use.

Several existing parameter names have been changed as part of this release. These new names were chosen to improve readability of the returned event information.

Old Name	New Name
AttributeName	Attribute.Name
ContainerId	Container.Id
ContainerName	Container.Name
ElementName	Element.Name
ElementPath	Element.Path
PIPointDescriptor	PIPoint.Descriptor
PIPointEngineeringUnits	PIPoint.EngineeringUnits
PIPointExtendedDescriptor	PIPoint.ExtendedDescriptor
PIPointPointSource	PIPoint.PointSource
PIPointPointType	PIPoint.PointType
PropertyNullable	Property.IsNullable
TypeId	Type.Id
TypeVersion	Type.Version
TypePropertyFormat	Property.Format
TypePropertyIndex	Property
TypePropertyName	Property.Name
TypePropertyType	Property.Type
TypePropertyUom	Property.Uom

Breaking Change: Property Nullability Rules

OMF v1.2 introduces the concept of "implicitly nullable" property types/formats. An implicitly nullable property is a property which may receive a value of `null` even if the declared type does not include `"null"`. An

erratum for the OMF v1.1 specification has been published to include the same rule. The following property types/formats are now implicitly nullable:

- Properties of type `array`
- Properties of type `object`
- Properties of type `string`, except when format is `Date-Time`
  - This does not include `isIndex` or `isName` properties, which are not allowed to be null.

The following are expected changes in behavior as a result of this change:

- `STATIC DATA` messages that include `null` for implicitly nullable properties will no longer be rejected
- `DYNAMIC DATA` messages that include `null` for implicitly nullable properties will no longer be rejected
- The default value for non-`Date-Time string` properties has been changed to `null`. In previous versions of PI Web API OMF Services, the default value for a non-`Date-Time string` property that was not explicitly marked as nullable was `""` (an empty string).
  - When omitting a value for a non-`Date-Time string` property in a CREATE or UPDATE DATA message, the recorded value will now be `null`. In Data Archive, this results in a "No Data" digital state being recorded.

Breaking Change: Events

- Operations that previously returned `TypeMismatch` now return `TypeConflictsWithExisting`.

Non-Breaking Change: OMF v1.2 TYPE Version Demoted to Metadata

In OMF versions 1.0 and 1.1, TYPE versions were a first-class feature. The original intention of TYPE versions was to allow client applications to maintain multiple versions of the same TYPE. However, in practice, this made interacting with TYPEs more burdensome - especially with the introduction of new features like `reftypeid` and TYPE-to-TYPE links as part of OMF v1.2. As a result, in **OMF v1.2**, TYPE versions have been demoted to metadata describing the TYPE, and are no longer required when referencing a TYPE.

When porting a client application from OMF v1.0 or v1.1 to OMF v1.2, if the client depends on the TYPE version exactly matching (ex. during `STATIC DATA` or `LINK` messages), the client will experience a breaking change where requests that previously failed will now succeed. **Applications which do not use OMF v1.2 are not impacted.**

Non-Breaking Change: Event Names

Some existing events have been renamed for readability.

Old Name	New Name
AFAttributeConflictsWithExpected	AFAttributeConflictsWithExisting
ContainerAlreadyExists	ContainerConflictsWithExisting
DuplicateType	TypeAlreadyExists

Old Name	New Name
PIPointConflictsWithOverwrite	PIPointConflictsWithExisting
StaticDataDoesNotMatchExisting	StaticDataConflictsWithExisting
StaticInstanceWithSpecifiedIdAlreadyExists	StaticInstanceConflictsWithExisting
TypeAlreadyExists	TypeConflictsWithExisting

Schema Changes

Types

- `array`, `object`, and non-`Date-Time` `string` properties are now implicitly nullable. Omitting `"null"` when specifying a property of these types no longer prevents a value of `null` from being accepted for that property. This does not apply to any property with the `isIndex` or `isName` flag.
- The `minimum` and `maximum` property fields will appear as child AF Attribute Templates with the minimum/maximum limit property set.
- `ENUM` TYPEs have been introduced.

Containers

- PI Point Zero and Span attributes now contain values derived from the `minimum`/`maximum` of the associated TYPE property.
- PI Point Point Source attribute now contains the `CONTAINER` Data Source if one was supplied. If no Data Source was specified, the default Point Source `PIWebAPI_OMF` will be used.
- PI Points of type `Digital` may now be created. The PI Point Digital Set Name attribute will be set according to the `OmfRefTypeId` value when the `OmfRefTypeQualifier` value is `Digital-Enum`. If the `OmfRefTypeQualifier` value is `Non-Digital-Enum`, then the PI Point will be created using the type specified by the `OmfPropertyType` and `OmfPropertyFormat` keys.
- PI Point Step attribute now contains a value derived from `OmfPropertyInterpolationMode`. Because PI Web API does not yet support interpolation, this value is always the default mode associated with the property type and format.
- PI Point Extended Descriptor attribute changes:
  - The value associated with the `OmfSchemaVersion` key has been changed to to `2`.
  - Backslashes are now escaped with a preceding backslash.
  - The `OmfExplicitDataSource` key may now be present. When present, it indicates whether an explicit `CONTAINER` Data Source was specified.
  - The `OmfRefTypeId` key may now be present. When present, it indicates the PI Point is using the `reftypeid` feature. When the `OmfRefTypeId` key is present, the `OmfRefTypeQualifier` key must also be present.
  - The `OmfRefTypeQualifier` key may now be present. When present, it indicates the qualifier of the reference type referred to by the property's `reftypeid`. This key must be present if the `OmfRefTypeId` is present. Allowed values are:
    - `Digital-Enum` - Indicates that the PI Point is expected to use a Digital Set. The expected Digital Set is specified by the `OmfRefTypeId` key.

    - `Non-Digital-Enum` - Indicates that the PI point is expected to use the type and format specified by the `OmfPropertyType` and `OmfPropertyFormat` keys.
  - A property format of `date-time` is now serialized as `"Date-Time"`. In prior schema versions, a format of `date-time` was serialized as `"DateTime"` (note the missing hyphen).
  - The `OmfPropertySimple` key is now present in the PI Point's extended descriptor attribute. The value indicates whether this property is considered "simple". Previously, this value was inferred from the PI Point name. The representation of resources derived from a simple property may differ from non-simple properties (for example, PI Point names or AF Attribute names).
  - The `OmfPropertyInterpolation` key may now be present. When present, it indicates the interpolation mode this property is expected to use. When not present, the interpolation mode is inferred from the property type and format. Allowed values are:
    - `Continuous` - Indicates that the property is using the `continuous` interpolation mode.
    - `Discrete` - Indicates that the property is using the `discrete` interpolation mode.
  - The `OmfPropertyMinimum` and `OmfPropertyMaximum` keys may now be present. The values for these keys indicate whether a property has an explicit minimum or maximum. This is necessary because the Classic PI Point class defines default values for Span and Zero, and so the attribute values cannot be used to determine whether a minimum or maximum has been specified.

Data

- `array`, `object`, and non-`Date-Time` `string` properties are now implicitly nullable. Applications that consume OMF data can no longer rely on values of properties of these types to be non-`null`.
- The `minimum` and `maximum` property fields will appear as child AF Attributes with the minimum/maximum limit property set.

2021 SP1 - Detailed Changes (PI Web API OMF Services)

Expanded Feature: OMF version 1.2

PI Web API 2021 SP1 extends the partial support for version 1.2 of the OMF specification introduced in PI Web API 2021.

Support for the following OMF v1.2 features has been added:

- The `TYPE` message `extrapolation` property.
- The `TYPE` property `interpolation` property.
- The `CONTAINER` message `extrapolation` property.
- The following `CONTAINER` message `propertyOverrides` properties:
  - `interpolation`
- Partial support for the `DATA` message `properties` property.
  - Currently, the PI Web API only supports the `properties` property in combination with a `typeid`. The `typeid` cannot be omitted. When used, any additional properties added via the `properties` keyword will extend the specified `TYPE`.

The following OMF v1.2 features are not yet supported by the PI Web API:

- The TYPE message `baseTypeId` property.
- Non-ENUM TYPE messages that omit the `classification` property.
- The CONTAINER message `metadata` and `tags` properties.
- The following CONTAINER message `propertyOverrides` properties:
  - `name`
  - `description`
- Type-to-type LINK messages.

New Concept: Interpolation and Extrapolation Modes

PI Web API 2021 SP1 introduces support for the `interpolation` and `extrapolation` keywords in version 1.2 of the OMF specification. These keywords are used to control how data queries against the backend store should behave. The interpolation mode is used by the PI Web API to control the PI Point Step attribute value. The extrapolation mode is stored by the PI Web API, but is not used by the PI System.

For information on how the Step attribute value is calculated, see the section titled "PI Point Schema Version 3".

New Concept: Non-TYPE-Derived Properties

PI Web API 2021 SP1 introduces support for the DATA message `properties` keyword added in OMF version 1.2. The `properties` keyword allows a DATA message to define additional properties that will be used when storing a value. These properties support all the same features as a property defined as part of a TYPE, including the `reftypeid`, `minimum`, and `maximum` keywords. When the `properties` keyword appears alongside a `typeid`, any additional properties must be additive to the specified TYPE, and they must not overlap.

The PI Web API currently requires that a `typeid` be specified when using the `properties` keyword, even though the OMF specification does not make this requirement. This is because the PI Web API retrieves STATIC DATA instances by performing attribute value searches, and the backing AF Server may not support attribute value searches without specifying an AF Element Template. The PI Web API also requires that any additional properties are non-system properties: they must not have the `isName`, `isIndex`, or `isQuality` keywords set to `true`.

Breaking Change: Interpolation Mode

The `interpolation` keyword is used by the PI Web API to control the PI Point Step attribute value. In earlier PI Web API versions, a default interpolation mode was always used. Under certain conditions, the interpolation mode would be written to the PI Point extended descriptor; however, because interpolation mode was only used internally, these conditions could not be met when using the PI Web API. With the release of PI Web API 2021 SP1, the interpolation mode can now be controlled by the client OMF application. Interpolation modes in the extended descriptor of PI Points using earlier schema versions will now be ignored for migration purposes.

If your application does not directly read the interpolation mode from PI Point extended descriptors, you are not affected by this change. This covers applications that only use OMF. If your application does read the interpolation mode, your application must be updated to use PI Point schema version 3.

Breaking Change: PI Point Schema Version 3

As part of PI Web API 2019 SP1's resource migration support, the concept of schema versions was introduced. Schema versions describe the structure of PI System resources that are created or interacted with by the PI Web API; schema versions are not part of the OMF specification. PI Web API 2021 SP1 introduces a new PI Point schema version: schema version 3. Schema version 3 introduces greater control over PI Points created using OMF, and supports new OMF features such as `interpolation` and `extrapolation`.

PI Points will automatically be upgraded to schema version 3 when a `CREATE CONTAINER` or `UPDATE CONTAINER` message is received. PI Points are upgraded to schema version 3 regardless of the version of OMF being used. OMF client application behavior is not impacted; no changes are required. PI System applications may experience changes in behavior (ex. value queries returning different results due to the PI Point Step attribute); for this reason, this change is considered a breaking change.

Each PI Point maps to a value property on the `TYPE` used by a `CONTAINER`. PI Point attributes are mapped using schema version 3 as follows:

PI Point Attribute	OMF Mapping
Name *(AKA Tag)*	• If the `CONTAINER` type is simple (has one value property), of the form `$"{container.Id}"`. • If the `CONTAINER` type is complex (has more than one value property), of the form `$"{container.Id}.{property.Index}"`. For more details on simple and complex types, see the PI Web API OMF Services 2019 SP1 detailed changes topic.
Descriptor	The `CONTAINER`'s `description`. If the `CONTAINER` does not have a `description`, this attribute is not set.
Point Source	• If the `CONTAINER` specifies a `dataSource`, the specified `dataSource`. • If the `CONTAINER` does not specify a `dataSource`, the default value `PIWebAPI_OMF`.
Point Type	• If the property is not using `reftypeid`, derived from the property's `type` and `format` according to the table below. • If the property is using `reftypeid`, if the referenced `ENUM` corresponds to a Digital Set,

`Digital`. Otherwise, derived from the `ENUM`'s `type` and `format` according to the table below.

Type	Format	PI Point Type	Notes
`array`		*Unsupported*	
`boolean`		`Int16`	
`integer`	`int64`	• If this PI Point is using `reftypeid`, `String` • If this PI Point is not using `reftypeid`, `Float64`	The Data Archive does not have a numerical data type large enough to store an `int64` without precision loss.
`integer`	`int32` *(Default - non-*`ENUM`*)*	`Int32`	
`integer`	`int16` *(Default -* `ENUM`*)*	`Int32`	The Data Archive has special behavior for `Int16` PI Points. For consistency, the PI Web API instead uses `Int32`.
`integer`	`uint64`	• If this PI Point is using	The Data Archive does not have a

		`refty peid`, `Strin g` • If this PI Point is not using `refty peid`, `Float 64`	numerical data type large enough to store a `uint64` without precision loss.
`integer`	`uint32`	`Float64`	The Data Archive does not have a native unsigned integer data type. However, a `Float64` can store a `uint32` without precision loss.
`integer`	`uint16`	`Int32`	The Data Archive does not have a native unsigned integer data type. However, an `Int32` can store a `uint16` without precision loss.
`number`	`float64`	`Float64`	
`number`	`float32` *(Default)*	`Float32`	

<table>
<tr><td></td><td>
<table>
<tr><td>number</td><td>float16</td><td>Float16</td><td>Although the Data Archive supports Float16, because the Asset Framework server does not, properties of type float16 cannot be created.</td></tr>
<tr><td>object</td><td>dictionary</td><td>Unsupported</td><td></td></tr>
<tr><td>string</td><td></td><td>String</td><td></td></tr>
<tr><td>string</td><td>date-time</td><td>Timestamp</td><td></td></tr>
</table>
</td></tr>
<tr><td>Digital Set Name</td><td>

If the property is using reftypeid and the specified ENUM is eligible for a Digital Set, the Digital Set with the name matching the ENUM's ID.
If the property is using reftypeid and the specified ENUM is not eligible for a Digital Set, this attribute is not set.
If the property is not using reftypeid, this attribute is not set.

</td></tr>
<tr><td>Eng Units</td><td>In order of precedence,

1. If the CONTAINER contains a property override for this property that specifies a uom, the specified uom.
2. If the property specifies a uom, the property's uom.
3. This attribute is not set.</td></tr>
<tr><td>Zero</td><td>In order of precedence,

1. If the CONTAINER contains a property override for this property that specifies a minimum, the specified minimum.</td></tr>
</table>

<table>
<tr><td></td><td>2. If the property specifies a <code>minimum</code>, the property's <code>minimum</code>.
3. The <code>Classic</code> PI Point Class' default zero.</td></tr>
<tr><td>Span</td><td>In order of precedence,

1. If the <code>CONTAINER</code> contains a property override for this property that specifies a <code>maximum</code>, a value derived from the specified <code>maximum</code>.
2. If the property specifies a <code>maximum</code>, a value derived from the property's <code>maximum</code>.
3. The <code>Classic</code> PI Point Class' default span.</td></tr>
<tr><td>Step</td><td>In order of precedence,

1. If the <code>CONTAINER</code> contains a property override for this property that specifies an <code>interpolation</code>, a value derived from the specified <code>interpolation</code>.
2. If the property specifies an <code>interpolation</code>, a value derived from the property's <code>interpolation</code>.
3. A default value derived according to the property's <code>type</code> and <code>format</code>.

The <code>interpolation</code> modes defined by the OMF specification are mapped to Step values as:
<table><tr><th>Interpolation Mode</th><th>Step Value</th><th>Notes</th></tr><tr><td><code>Continuous</code></td><td><code>0</code></td><td></td></tr><tr><td><code>StepwiseContinuousLeading</code></td><td><code>0</code></td><td>The Data Archive doesn't natively support this interpolation mode. Because a Step of <code>1</code> would misleadingly cause interpolated values to behave as stepwise following, the PI</td></tr></table></td></tr>
</table>

		Web API treats this interpolation mode as continuous.
`Discrete`	`1`	The Data Archive doesn't natively support this interpolation mode. The PI Web API treats this interpolation mode as stepwise following because it is the most similar natively supported mode.
`StepwiseContinuousFollowing`	`1`	

The following PI Point Types do not support interpolation. If a PI Point is using any of these types, the `interpolation` and property `type` and `format` are ignored.

Point Type	Step Value
`String`	`1`
`Timestamp`	`1`
`Blob`	`1`
`Digital`	`1`

The following property `type` and `format` combinations do not support interpolation, and will override any specified `interpolation`:

Type	Format	Step Value
`array`		*Unsupported*
`boolean`		`1`
`object`	`dictionary`	*Unsupported*
`string`		`1`

<table>
<tr><td></td><td>
<table>
<tr><td>string</td><td>date-time</td><td>1</td></tr>
</table>
If no interpolation is specified, if reftypeid is being used, the default Step value is 1. Otherwise, the default Step value will be:
<table>
<tr><th>Type</th><th>Format</th><th>Step Value</th></tr>
<tr><td>array</td><td></td><td>Unsupported</td></tr>
<tr><td>boolean</td><td></td><td>1</td></tr>
<tr><td>integer</td><td>int64</td><td>0</td></tr>
<tr><td>integer</td><td>int32 (Default - non-ENUM)</td><td>0</td></tr>
<tr><td>integer</td><td>int16 (Default - ENUM)</td><td>0</td></tr>
<tr><td>integer</td><td>uint64</td><td>0</td></tr>
<tr><td>integer</td><td>uint32</td><td>0</td></tr>
<tr><td>integer</td><td>uint16</td><td>0</td></tr>
<tr><td>number</td><td>float64</td><td>0</td></tr>
<tr><td>number</td><td>float32 (Default)</td><td>0</td></tr>
<tr><td>number</td><td>float16</td><td>0</td></tr>
<tr><td>object</td><td>dictionary</td><td>Unsupported</td></tr>
<tr><td>string</td><td></td><td>1</td></tr>
<tr><td>string</td><td>date-time</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Extended Descriptor</td><td>The extended descriptor is reserved for internal PI Web API use.</td></tr>
</table>

Non-Breaking Change: Event Names

Some existing events have been renamed for readability.

Old Name	New Name
ElementContainsUnrecognizedOmfAttribute	ElementContainsUnknownExtensionAttribute
SpecifiedOverrideIsNotValidForProperty	SpecifiedOverridesAreNotValidForProperty
TypeDoesNotHavePropertyWithSpecifiedIndex	PropertyWithSpecifiedIndexNotFound

Informational: Migrating from Prior PI Web API Versions

AF Attributes Created Via LINK Messages

If a `minimum` or `maximum` override is added to a `CONTAINER` while the PI Web API is being upgraded from 2021 to 2021 SP1, existing AF Attributes created via `LINK` messages may incorrectly have the `minimum` and `maximum` added as child attributes, rather than emit a mismatch event.

Schema Changes

Types

- The `interpolation` field will appear as a child AF Attribute Template.
- The `extrapolation` field will appear as an extended property of the AF Element Template.

Containers

- PI Point Step attribute now contains a value derived from `OmfPropertyInterpolationMode`.
- PI Point Extended Descriptor attribute changes:
  - The value associated with the `OmfSchemaVersion` key has been changed to to `3`.
  - The `OmfPropertyInterpolation` key may now be present. When present, it indicates the interpolation mode this property is expected to use. When not present, it indicates no explicit interpolation mode was specified. Allowed values are:
    - `Continuous` - Indicates that the property is using the `continuous` interpolation mode.
    - `Discrete` - Indicates that the property is using the `discrete` interpolation mode.
    - `StepwiseContinuousLeading` - Indicates that the property is using the `stepwisecontinuousleading` interpolation mode.
    - `StepwiseContinuousFollowing` - Indicates that the property is using the `stepwisecontinuousfollowing` interpolation mode.
  - The `OmfContainerExtrapolationMode` key may now be present. When present, it indicates the extrapolation mode this property is expected to use. When not present, it indicates no explicit extrapolation mode was specified. Allowed values are:
    - `All` - Indicates that the property is using the `all` extrapolation mode.
    - `None` - Indicates that the property is using the `none` extrapolation mode.
    - `Forward` - Indicates that the property is using the `forward` extrapolation mode.
    - `Backward` - Indicates that the property is using the `backward` extrapolation mode.
- When reading a PI Point using schema version `2`, the PI Web API will ignore the `OmfPropertyInterpolation` key if it is present.

Data

- Non-TYPE-derived properties will appear as AF Attributes.
  - If a non-TYPE-derived property has a `minimum`, `maximum`, or `interpolation`, it will be stored as child AF Attributes in the same manner as a TYPE-derived property.

- When creating a `LINK` where a target PI Point has an Engineering Units attribute whose value does not correspond to an AF UOM, an AF Attribute will be created without a UOM.

2021 SP2 - Detailed Changes (PI Web API OMF Services)

New Feature: `Accept-Verbosity` header

PI Web API 2021 SP2 adds support for the `Accept-Verbosity` header when included on an OMF request. When the PI Web API processes an OMF request, the returned response may include information related to the request, such as any warnings or errors that occurred during processing. Previously, the PI Web API server could be configured to include additional information to assist with debugging client applications, and clients had no control over what information was included in the response. Now, by including the `Accept-Verbosity` header, a client application can indicate that it would like to receive additional information in the response regardless of the server-side configuration.

Note: The `Accept-Verbosity` header indicates that the client is prepared to receive this additional information. The additional information may include values supplied by the client, PI System internal state, or other information determined at runtime. This information is not included in responses by default due to the security risk it can pose to insecure clients. If a client includes the `Accept-Verbosity` header but does not handle the response securely (ex. by escaping returned strings before presenting them to the user, by preventing deserialization of the response from requesting external resources, etc.), the client may be exposed to reflected XSS attacks, information disclosure attacks, and other vulnerabilities. The safest way to avoid such attacks is to not include the `Accept-Verbosity` header on requests. However, if the design of your application requires this information, be aware of the risks and ensure the returned information is handled correctly.

The `Accept-Verbosity` header can be used with the following values:

Header Value	Additional Details Returned	Description
`parameters`	Parameters associated with log events	Events may include parameters that help contextualize the event. For example, if an event says a TYPE was not found, it may include a parameter indicating the ID of the missing TYPE.
`exceptions`	Exception information associated with a log event	Events may arise as a result of an exception occurring during processing. If an event has such an exception associated with it, it will include information about the exception that occurred. For example, if an AF or DA operation failed due to a runtime exception, the exception information may help diagnose the underlying cause of the failure.
`inner-events`	Inner log events contained within a parent log event	Events may include inner events that help contextualize the event.

		For example, if an event indicates a conflict with an existing PI Point, the inner events may contain information describing the reason the conflict occurred.
`verbose`	Behaves the same as specifying both `parameters` and `inner-events`.	

Multiple `Accept-Verbosity` headers and/or values can be sent within the same request. For example, a client could indicate that the response should include event parameters and exceptions in either of the following ways:

`Accept-Verbosity` header specified multiple times:

```
Accept-Verbosity: parameters
Accept-Verbosity: exceptions
```

`Accept-Verbosity` header specified once, with multiple values:

```
Accept-Verbosity: parameters, exceptions
```

Breaking Change: 'InnerEvents' and 'OmfIncludeInnerEvents' configuration option

When the PI Web API receives an OMF request, it will respond with a series of events that occurred during processing. These events can include various types of information, such as errors, warnings, or informational messages. These events may themselves be composed of inner child events that give greater context to the parent event. Previously, these inner events would only be included in a PI Web API OMF response when the PI Web API had been configured to run in Debug Mode. This meant that inner events were not available in production environments. PI Web API 2021 SP2 introduces a new configuration setting, `OmfIncludeInnerEvents`, which can be used to enable or disable inner events in production environments.

By default, `OmfIncludeInnerEvents` is set to `true`, and inner events will be returned. For backwards compatibility, returning inner events can be disabled by setting the `OmfIncludeInnerEvents` option to `false`. For more instructions on runtime configuration, see the PI Web API Admin Guide.

2021 SP3 - Detailed Changes (PI Web API OMF Services)

Non-Breaking Change: Simplified `DELETE` for `TYPE`, `CONTAINER`, and Static `DATA`

PI Web API 2021 SP3 implements simplified handling of `DELETE` messages when deleting Types, Containers, and Static Data. The table below shows the updated required fields for `DELETE` processing.

Target Entity	Required Fields	Example Message Body
TYPE	id	[{"id":"TypeIdToDelete"}]
CONTAINER	id typeid	[{"id": "ContainerIdToDelete", "typeid": "Widget"}]
STATIC DATA	typeid values[{index}]	[{ "typeid":"Widget", "values": [{ "WidgetId": "WidgetToDelete" }] }] Where WidgetId is the index property of the TYPE

If any other fields are provided in the DELETE message, they will be ignored. This change relaxes the requirement that PI Web API clients persist the full entity definitions to be able to delete them.

Note: Previously a successful DELETE message required that all fields were present and matched the entity definition. With this simplified behavior, the entity will be deleted as long as it was found given the required fields provided. For the message types referenced above, there is no longer a case that will return a mismatch error, e.g. "A type with the supplied ID and version already exists, but it does not match the supplied type."

Non-Breaking Change: Case Insensitivity for Type Id, Container Id, and Static Index

PI Web API 2021 SP3 relaxes the requirement that a provided Type Id, Container Id, or Static Index match in case with the value in the entity definition. For example, if a Static Type is defined with id WidgetType, a client can create a new static data instance of that type with the message field "typeid": "widgettype". This new behavior applies to all id, typeid, containerid, and index fields.

Note: In some rare cases an ambiguity may arise if the provided field value does not exactly match an existing entity, but does case-insensitvely match multiple entities. In that event, PI Web API will proceed with the first entity found and will return a warning level message in the response indicating an ambigous action was taken, e.g. "More than one static instance with the specified ID was found. Affected instance may not be the intended one." The exception to this is during DELETE operations in which case an error is returned and no action is taken.

Non-Breaking Change: Warning on Precision Loss

Precision loss may occur when converting values in OMF messages to their corresponding data types as defined in their type definition. For example, a decimal value with more digits than are supported by the float32 format. Previously this would result in an error status which could halt execution. In PI Web API 2021 SP3, precision loss detected while processing data values or property min/max values will instead result in a warning in the repsonse, but the action will proceed.

Non-Breaking Change: Container CREATE/UPDATE Does Not Overwrite Step If Not Specified

When creating or updating a CONTAINER, if the PI Point already exists on the server PI Web API will no longer overwrite the existing Step attribute of the point unless an interpolation value is specified in the message.

Non-Breaking Change: Container UPDATE Does Not Overwrite DataSource If Not Specified

When updating a CONTAINER, if the datasource field is not present in the message, PI Web API will no longer overwrite the existing PointSource attribute of the corresponding PI Point(s).

OMF Endpoint Notes (PI Web API OMF Services)

The Open Message Format (OMF) defines a format for data messages that can be read by compliant OSIsoft products. It defines an abstract message type, where a message consists of a set of key/value pairs, called the header, and a binary payload, called the body. It defines how the messages should be constructed and what they can contain. OMF messages can be constructed using any message protocol that defines headers and bodies. See the Open Message Format specification for more information about the Open Message Format.

The Developer Companion Guide is specific to the PI System and describes how to use OMF in your application to send data to PI System destinations, including PI AF Server and PI Data Archive. It includes:

- How different message types are interpreted by the PI System
- OMF event codes
- Restrictions and limitations
- Development considerations
- Getting started information
- Information on migrating existing OMF applications to OMF with PI Web API

Additional resources are available to help you get started with OMF:

- OMF on PI Square
- OMF Samples

OMF Performance Metrics (PI Web API OMF Services)

This topic is only a brief overview of OMF Services-specific performance metrics. For more information regarding the Performance Metrics feature, see the Performance Metrics topic.

PI Web API OMF Services Performance Metrics include:

- **Operation metrics** - Returns OMF version-specific metrics. The set of metrics may differ based on the OMF version. Metrics are grouped by their associated OMF version.

  - **OMF v1.0** - The PI Web API aliases OMF v1.0 messages to OMF v1.1. Messages using OMF v1.0 will appear under the OMF v1.1 metric group.
  - **OMF v1.1** - Returns counts of:
    - Type messages
    - Container messages
    - Static data messages
    - Dynamic data messages
    - Link data messages
  - **OMF v1.2** - Returns counts of:
    - Type messages
    - Container messages
    - Static data messages
    - Dynamic data messages
    - Link data messages

- **PI Data Archive throughput** - For each DA server, returns counts of:
  - Points created
  - Points updated
  - Points deleted
  - Values inserted
  - Values updated
  - Values removed
  - Digital sets created
  - Digital sets deleted
- **Asset Framework throughput** - For each AF database, returns counts of:
  - Element templates created
  - Element templates deleted
  - Elements created
  - Elements updated
  - Elements deleted
  - Elements moved
  - Enumeration sets created
  - Enumeration sets deleted
  - Point reference attributes created
  - Point reference attributes updated
  - Point reference attributes deleted

PI Web API OMF Migration (PI Web API OMF Services)

When migrating from a previous version of the PI Web API OMF Services, you should be aware of the following:

- OMF resources may have had their PI AF or PI Data Archive representations changed.
- Existing PI AF or PI Data Archive resources that were not created using OMF may be migrated.

For additional information, refer to the Developer Companion Guide.

Details

- PI Web API OMF Services 2021 SP3
- PI Web API OMF Services 2021 SP2
- PI Web API OMF Services 2021 SP1
- PI Web API OMF Services 2021
- PI Web API OMF Services 2019 SP1
- PI Web API OMF Services 2019

WebID Encoded Datatypes (WebID)

For all WebID instances, the first character is an indicator of the Web ID type (see: WebID Type). The second character is a version character. The third and fourth characters are a two-character marker indicating the datatype to which the identifier refers. Name payloads are converted to uppercase, so that Web ID generation is case-insensitive. (see: WebID Encoding)

Table of Contents

AF Analysis

The Web ID for an AFAnalysis consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Xs'	2	None
System ID	analysis.PISystem.ID	22	Urlencoded Guid
Analysis ID	analysis.ID	22	Urlencoded Guid
Name Payload	`analysis.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Xs'	2	None
System ID	analysis.PISystem.ID	22	Urlencoded Guid
Analysis ID	analysis.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Xs'	2	None
Name Payload	`analysis.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Xs'	2	None

Field Name	Value	Encoded Width	Encoding Method
Analysis ID	analysis.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'Xs'	2	None
Analysis ID	analysis.ID	22	Urlencoded Guid

Back to Table of Contents

AF AnalysisRule

The Web ID for an AFAnalysisRule consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'XR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
System ID	analysisRule.PISystem.ID	22	Urlencoded Guid
Owner ID	analysisRule.Analysis.ID or analysisRule.AnalysisTemplate.ID	22	Urlencoded Guid
Analysis Rule ID	analysisRule.ID	22	Urlencoded Guid
Name Payload	`analysisRule.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'XR'	2	None

Field Name	Value	Encoded Width	Encoding Method
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
System ID	analysisRule.PISystem.ID	22	Urlencoded Guid
Owner ID	analysisRule.Analysis.ID or analysisRule.AnalysisTemplate.ID	22	Urlencoded Guid
Analysis Rule ID	analysisRule.category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'XR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Name Payload	`analysisRule.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'XR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Owner ID	analysisRule.Analysis.ID or analysisRule.AnalysisTemplate.ID	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
Analysis Rule ID	analysisRule.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'XR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Owner ID	analysisRule.Analysis.ID or analysisRule.AnalysisTemplate.ID	22	Urlencoded Guid
Analysis Template ID	template.ID	22	Urlencoded Guid

Back to Table of Contents

AFAnalysisTemplate

The Web ID for an AFAnalysisTemplate consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'XT'	2	None
System ID	template.PISystem.ID	22	Urlencoded Guid
Analysis Template ID	template.ID	22	Urlencoded Guid
Name Payload	`template.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'XT'	2	None

Field Name	Value	Encoded Width	Encoding Method
System ID	template.PISystem.ID	22	Urlencoded Guid
Analysis Template ID	template.category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'XT'	2	None
Name Payload	`template.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'XT'	2	None
Analysis Template ID	template.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'XT'	2	None
Analysis Template ID	template.ID	22	Urlencoded Guid

Back to Table of Contents

AFAttribute

The Web ID for an AFAttribute consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Ab'	2	None
Base Element Marker	• 'E' if attr.Element is AFElement	1	None

Field Name	Value	Encoded Width	Encoding Method
	• 'F' if attr.Element is AFEventFrame • 'N' if attr.Element is AFNotification		
System ID	attr.PISystem.ID	22	Urlencoded Guid
Element ID	attr.Element.ID	22	Urlencoded Guid
Attribute ID	attr.ID	22	Urlencoded Guid
Name Payload	`attr.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Ab'	2	None
Base Element Marker	• 'E' if attr.Element is AFElement • 'F' if attr.Element is AFEventFrame • 'N' if attr.Element is AFNotification	1	None
System ID	attr.PISystem.ID	22	Urlencoded Guid
Element ID	attr.Element.ID	22	Urlencoded Guid
Attribute ID	attr.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Ab'	2	None
Base Element Marker	• 'E' if attr.Element is AFElement • 'F' if attr.Element is AFEventFrame	1	None

Field Name	Value	Encoded Width	Encoding Method
	• 'N' if attr.Element is AFNotification		
Name Payload	`attr.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Ab'	2	None
Base Element Marker	• 'E' if attr.Element is AFElement • 'F' if attr.Element is AFEventFrame • 'N' if attr.Element is AFNotification	1	None
Element ID	attr.Element.ID	22	Urlencoded Guid
Attribute ID	attr.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'Ab'	2	None
Event Frame ID	analysisRulePlugIn.ID	22	Urlencoded Guid
Base Element Marker	• 'E' if attr.Element is AFElement • 'F' if attr.Element is AFEventFrame • 'N' if attr.Element is AFNotification	1	None
Element ID	attr.Element.ID	22	Urlencoded Guid
Attribute ID	attr.ID	22	Urlencoded Guid

Back to Table of Contents

AFAttributeTemplate

The Web ID for an AFAttributeTemplate consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'AT'	2	None
Element Template Marker	'E' *Preserved here for legacy purposes*	1	None
System ID	attrTempl.PISystem.ID	22	Urlencoded Guid
Element Template ID	attrTempl.ElementTemplate.ID	22	Urlencoded Guid
Attribute Template ID	attrTempl.ID	22	Urlencoded Guid
Name Payload	`attrTempl.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'AT'	2	None
Element Template Marker	'E' *Preserved here for legacy purposes*	1	None
System ID	attrTempl.PISystem.ID	22	Urlencoded Guid
Element Template ID	attrTempl.ElementTemplate.ID	22	Urlencoded Guid
Attribute Template ID	attrTempl.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'AT'	2	None
Element Template Marker	'E' *Preserved here for legacy purposes*	1	None
Name Payload	`attrTempl.GetPath(AFEncodeType.Na`	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	me, null).Substring(2).ToUpperInvariant()		

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'AT'	2	None
Element Template Marker	'E' *Preserved here for legacy purposes*	1	None
Element Template ID	attrTempl.ElementTemplate.ID	22	Urlencoded Guid
Attribute Template ID	attrTempl.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'AT'	2	None
Element Template Marker	'E' *Preserved here for legacy purposes*	1	None
Element Template ID	attrTempl.ElementTemplate.ID	22	Urlencoded Guid
Attribute Template ID	attrTempl.ID	22	Urlencoded Guid

Back to Table of Contents

AFCategory for Analysis

The Web ID for an AFCategory, whose Identity is CategoryAnalysis, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'XC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Analysis Template ID	category.ID	22	Urlencoded Guid
Name Payload	category.GetPath(AFEncodeType.Name	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	, null).Substring(2).ToUpperInvariant()		

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'XC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Analysis Template ID	category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'XC'	2	None
Name Payload	category.GetPath(AFEncodeType.Name , null).Substring(2).ToUpperInvariant()	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'XC'	2	None
Analysis Template ID	category.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'XC'	2	None
Analysis Template ID	category.ID	22	Urlencoded Guid

Back to Table of Contents

AFCategory for Attribute

The Web ID for an AFCategory, whose Identity is CategoryAttribute, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'AC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'AC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'AC'	2	None
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'AC'	2	None

Field Name	Value	Encoded Width	Encoding Method
Category ID	category.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'AC'	2	None
Category ID	category.ID	22	Urlencoded Guid

Back to Table of Contents

AFCategory for Element

The Web ID for an AFCategory, whose Identity is CategoryElement, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'EC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'EC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'EC'	2	None

Field Name	Value	Encoded Width	Encoding Method
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'EC'	2	None
Category ID	category.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'EC'	2	None
Category ID	category.ID	22	Urlencoded Guid

Back to Table of Contents

AFCategory for Table

The Web ID for an AFCategory, whose Identity is CategoryTable, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'BC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'BC'	2	None
System ID	category.PISystem.ID	22	Urlencoded Guid
Category ID	category.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'BC'	2	None
Name Payload	`category.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'BC'	2	None
Category ID	category.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'BC'	2	None
Category ID	category.ID	22	Urlencoded Guid

Back to Table of Contents

AFDatabase

The Web ID for an AFDatabase consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None

Field Name	Value	Encoded Width	Encoding Method
Marker	'RD'	2	None
System ID	database.PISystem.ID	22	Urlencoded Guid
Database ID	database.ID	22	Urlencoded Guid
Name Payload	`database.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'RD'	2	None
System ID	database.PISystem.ID	22	Urlencoded Guid
Database ID	database.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'RD'	2	None
Name Payload	`database.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'RD'	2	None
Database ID	database.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'RD'	2	None

Field Name	Value	Encoded Width	Encoding Method
Database ID	database.ID	22	Urlencoded Guid

Back to Table of Contents

AFElement

The Web ID for an AFElement consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Em'	2	None
System ID	element.PISystem.ID	22	Urlencoded Guid
Element ID	element.ID	22	Urlencoded Guid
Name Payload	`element.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Em'	2	None
System ID	element.PISystem.ID	22	Urlencoded Guid
Element ID	element.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Em'	2	None
Name Payload	`element.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Em'	2	None
Element ID	element.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'Em'	2	None
Element ID	element.ID	22	Urlencoded Guid

Back to Table of Contents

AFElementTemplate

The Web ID for an AFElementTemplate consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'ET'	2	None
System ID	template.PISystem.ID	22	Urlencoded Guid
Element Template ID	template.ID	22	Urlencoded Guid
Name Payload	`template.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'ET'	2	None
System ID	template.PISystem.ID	22	Urlencoded Guid
Element Template ID	template.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'ET'	2	None
Name Payload	`template.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'ET'	2	None
Element Template ID	template.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'ET'	2	None
Element Template ID	template.ID	22	Urlencoded Guid

Back to Table of Contents

AFEnumerationSet

The Web ID for an AFEnumerationSet consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'MS'	2	None
Source Marker	• 'R' if `enumerationSet.PISystem != null` • 'D' if `enumerationSet`	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
	`.PIServer != null`		
Server ID	enumerationSet.PISystem.ID (if AF), enumerationSet.PIServer.ID (if PI)	22	Urlencoded Guid
Enumeration Set ID	enumerationSet.ID	22	Urlencoded Guid
Name Payload	`enumerationSet.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'MS'	2	None
Source Marker	• 'R' if `enumerationSet.PISystem != null` • 'D' if `enumerationSet.PIServer != null`	22	Urlencoded Guid
Server ID	• enumerationSet.PISystem.ID (if AF) • enumerationSet.PIServer.ID (if PI)	22	Urlencoded Guid
Enumeration Set ID	enumerationSet.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None

Field Name	Value	Encoded Width	Encoding Method
Marker	'MS'	2	None
Source Marker	• 'R' if `enumerationSet.PISystem != null` • 'D' if `enumerationSet.PIServer != null`	22	Urlencoded Guid
Name Payload	`enumerationSet.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'MS'	2	None
Source Marker	• 'R' if `enumerationSet.PISystem != null` • 'D' if `enumerationSet.PIServer != null`	22	Urlencoded Guid
Enumeration Set ID	enumerationSet.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'MS'	2	None
Source Marker	• 'R' if `enumerationSet`	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
	.PISystem != null • 'D' if enumerationSet.PIServer != null		
Enumeration Set ID	enumerationSet.ID	22	Urlencoded Guid

Back to Table of Contents

AFEnumerationValue

The Web ID for an AFEnumerationValue consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'MV'	2	None
Source Marker	• 'R' if val.EnumerationSet.PISystem != null • 'D' if val.EnumerationSet.PIServer != null	22	Urlencoded Guid
Server ID	• val.EnumerationSet.PISystem.ID (if AF) • val.EnumerationSet.PIServer.ID (if PI)	22	Urlencoded Guid
Enumeration Set ID	val.EnumerationSet.ID	22	Urlencoded Guid
Enumeration Value ID	val.ID	22	Urlencoded Guid
Name Payload	(val.EnumerationSet.GetPath(AFEncodeType.Name, null).Substring(2	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	`) + '\\' + val.Name) .ToUpperInvariant()`		

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'MV'	2	None
Source Marker	• 'R' if `val.EnumerationSet.PISystem != null` • 'D' if `val.EnumerationSet.PIServer != null`	22	Urlencoded Guid
Server ID	• val.EnumerationSet.PISystem.ID (if AF) • val.EnumerationSet.PIServer.ID (if PI)	22	Urlencoded Guid
Enumeration Set ID	val.EnumerationSet.ID	22	Urlencoded Guid
Enumeration Value ID	val.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'MV'	2	None
Source Marker	• 'R' if `val.EnumerationSet.PISystem != null` • 'D' if `val.Enumeratio`	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
	`nSet.PIServer != null`		
Name Payload	`(val.EnumerationSet.GetPath(AFEncodeType.Name, null).Substring(2) + '\\' + val.Name) .ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'MV'	2	None
Source Marker	• 'R' if `val.EnumerationSet.PISystem != null` • 'D' if `val.EnumerationSet.PIServer != null`	22	Urlencoded Guid
Enumeration Set ID	val.EnumerationSet.ID	22	Urlencoded Guid
Enumeration Value ID	val.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'MV'	2	None
Source Marker	• 'R' if `val.EnumerationSet.PISystem != null` • 'D' if `val.EnumerationSet.PIServer != null`	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
Enumeration Set ID	val.EnumerationSet.ID	22	Urlencoded Guid
Enumeration Value ID	val.ID	22	Urlencoded Guid

Back to Table of Contents

AFEventFrame

The Web ID for an AFEventFrame consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Fm'	2	None
System ID	eventFrame.PISystem.ID	22	Urlencoded Guid
Event Frame ID	eventFrame.ID	22	Urlencoded Guid
Name Payload	`eventFrame.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Fm'	2	None
System ID	eventFrame.PISystem.ID	22	Urlencoded Guid
Event Frame ID	eventFrame.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Fm'	2	None
Name Payload	`eventFrame.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Fm'	2	None
Event Frame ID	eventFrame.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'Fm'	2	None
Event Frame ID	eventFrame.ID	22	Urlencoded Guid

Back to Table of Contents

AFNotification

The Web ID for an AFNotification consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Nf'	2	None
System ID	notification.PISystem.ID	22	Urlencoded Guid
Element ID	notification.ID	22	Urlencoded Guid
Name Payload	`notification.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Nf'	2	None
System ID	notification.PISystem.ID	22	Urlencoded Guid
Element ID	notification.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Nf'	2	None
Name Payload	`notification.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Nf'	2	None
Element ID	notification.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'Nf'	2	None
Element ID	notification.ID	22	Urlencoded Guid

Back to Table of Contents

AFNotificationtemplate

The Web ID for an AFNotificationTemplate consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'NT'	2	None
System ID	template.PISystem.ID	22	Urlencoded Guid
Element Template ID	template.ID	22	Urlencoded Guid
Name Payload	`template.GetPath(AFEncodeType.Name, null).Substring(2`	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	).ToUpperInvariant()		

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'NT'	2	None
System ID	template.PISystem.ID	22	Urlencoded Guid
Element Template ID	template.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'NT'	2	None
Name Payload	template.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'NT'	2	None
Element Template ID	template.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'NT'	2	None
Element Template ID	template.ID	22	Urlencoded Guid

[Back to Table of Contents](#toc)

AFPlugIn for Analysis Rule

The Web ID for an AFPlugIn, whose Identity is PlugInAnalysisRule, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'XP'	2	None
System ID	analysisRulePlugIn.PISystem.ID	22	Urlencoded Guid
Event Frame ID	analysisRulePlugIn.ID	22	Urlencoded Guid
Name Payload	`analysisRule.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'XP'	2	None
System ID	analysisRulePlugIn.PISystem.ID	22	Urlencoded Guid
Event Frame ID	analysisRulePlugIn.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'XP'	2	None
Name Payload	`analysisRulePlugIn.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'XP'	2	None
Event Frame ID	analysisRulePlugIn.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'XP'	2	None
Event Frame ID	analysisRulePlugIn.ID	22	Urlencoded Guid

Back to Table of Contents

AFPlugIn for Time Rule

The Web ID for an AFPlugIn, whose Identity is PlugInTimeRule, consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'TP'	2	None
System ID	timeRulePlugIn.PISystem.ID	22	Urlencoded Guid
Event Frame ID	timeRulePlugIn.ID	22	Urlencoded Guid
Name Payload	`timeRulePlugIn.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'TP'	2	None
System ID	timeRulePlugIn.PISystem.ID	22	Urlencoded Guid
Event Frame ID	timeRulePlugIn.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'TP'	2	None

Field Name	Value	Encoded Width	Encoding Method
Name Payload	`timeRulePlugIn.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'TP'	2	None
Event Frame ID	timeRulePlugIn.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'TP'	2	None
Event Frame ID	timeRulePlugIn.ID	22	Urlencoded Guid

Back to Table of Contents

AFSecurityIdentity

The Web ID for an AFSecurityIdentity consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'SI'	2	None
System ID	securityIdentity.PISystem.ID	22	Urlencoded Guid
Security Identity ID	securityIdentity.ID	22	Urlencoded Guid
Name Payload	`securityIdentity.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'SI'	2	None
System ID	securityIdentity.PISystem.ID	22	Urlencoded Guid
Security Identity ID	securityIdentity.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'SI'	2	None
Name Payload	`securityIdentity.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'SI'	2	None
Security Identity ID	securityIdentity.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'SI'	2	None
Security Identity ID	securityIdentity.ID	22	Urlencoded Guid

Back to Table of Contents

AFSecurityMapping

The Web ID for an AFSecurityMapping consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'SM'	2	None
System ID	securityMapping.PISystem.ID	22	Urlencoded Guid
Security Mapping ID	securityMapping.ID	22	Urlencoded Guid
Name Payload	`securityMapping.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'SM'	2	None
System ID	securityMapping.PISystem.ID	22	Urlencoded Guid
Security Mapping ID	securityMapping.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'SM'	2	None
Name Payload	`securityMapping.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'SM'	2	None
Security Mapping ID	securityMapping.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'SM'	2	None
Security Mapping ID	securityMapping.ID	22	Urlencoded Guid

Back to Table of Contents

AFTable

The Web ID for an AFTable consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'BI'	2	None
System ID	table.PISystem.ID	22	Urlencoded Guid
Table ID	table.ID	22	Urlencoded Guid
Name Payload	`table.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'BI'	2	None
System ID	table.PISystem.ID	22	Urlencoded Guid
Table ID	table.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'BI'	2	None
Name Payload	`table.GetPath(AFEncodeType.Name, null).Substring(2`	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	).ToUpperInvariant()		

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'BI'	2	None
Table ID	table.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'BI'	2	None
Table ID	table.ID	22	Urlencoded Guid

Back to Table of Contents

AFTimeRule

The Web ID for an AFTimeRule consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'TR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
System ID	timeRule.PISystem.ID	22	Urlencoded Guid
Owner ID	timeRule.Analysis.ID or timeRule.AnalysisTemplate.ID	22	Urlencoded Guid
Time Rule ID	timeRule.ID	22	Urlencoded Guid
Name Payload	timeRule.GetPath(AFEncodeType.Name, null).Substring(2	var	Urlencoded UTF8 String

Field Name	Value	Encoded Width	Encoding Method
	).ToUpperInvariant()		

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'TR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
System ID	timeRule.PISystem.ID	22	Urlencoded Guid
Owner ID	timeRule.Analysis.ID or timeRule.AnalysisTemplate.ID	22	Urlencoded Guid
Time Rule ID	timeRule.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'TR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Name Payload	timeRule.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'TR'	2	None

Field Name	Value	Encoded Width	Encoding Method
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Owner ID	timeRule.Analysis.ID or timeRule.AnalysisTemplate.ID	22	Urlencoded Guid
Time Rule ID	timeRule.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'TR'	2	None
Owner Marker	• 'X' if rule's owner is AFAnalysis • 'T' if rule's owner is AFAnalysisTemplate	1	None
Owner ID	timeRule.Analysis.ID or timeRule.AnalysisTemplate.ID	22	Urlencoded Guid
Time Rule ID	timeRule.ID	22	Urlencoded Guid

Back to Table of Contents

PIPoint

The Web ID for a PI Point consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'DP'	2	None
Server ID	point.Server.ID	22	Urlencoded Guid
Point ID	point.ID	6	Urlencoded 32-bit Integer
Name Payload	`point.GetPath().Substring(2) .ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'DP'	2	None
Server ID	point.Server.ID	22	Urlencoded Guid
Point ID	point.ID	6	Urlencoded 32-bit Integer

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'DP'	2	None
Name Payload	`point.GetPath().Substring(2) .ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'DP'	2	None
Point ID	point.ID	6	Urlencoded 32-bit Integer

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'DP'	2	None
Point ID	point.ID	6	Urlencoded 32-bit Integer

Back to Table of Contents

PIServer

The Web ID for a PI Server consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'DS'	2	None
Server ID	server.ID	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
Server Name	server.Name.ToUpperInvariant()	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'DS'	2	None
Server ID	server.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'DS'	2	None
Server Name	server.Name.ToUpperInvariant()	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'DS'	2	None

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'DS'	2	None

Back to Table of Contents

PISystem

The Web ID for a PISystem consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'RS'	2	None
System ID	piSystem.ID	22	Urlencoded Guid

Field Name	Value	Encoded Width	Encoding Method
Name Payload	`piSystem.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'RS'	2	None
System ID	piSystem.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'RS'	2	None
Name Payload	`piSystem.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'RS'	2	None

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'RS'	2	None

Back to Table of Contents

UOM

The Web ID for a UOM consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'Ut'	2	None
System ID	uom.PISystem.ID	22	Urlencoded Guid
UOM ID	uom.ID	22	Urlencoded Guid
Name Payload	`uom.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'Ut'	2	None
System ID	uom.PISystem.ID	22	Urlencoded Guid
UOM ID	uom.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'Ut'	2	None
Name Payload	`uom.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'Ut'	2	None
UOM ID	uom.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None

Field Name	Value	Encoded Width	Encoding Method
Marker	'Ut'	2	None
UOM ID	uom.ID	22	Urlencoded Guid

Back to Table of Contents

UOMClass

The Web ID for a UOMClass consists of the following data:

1. Full Type

Field Name	Value	Encoded Width	Encoding Method
Type	'F'	1	None
Version	'1'	1	None
Marker	'UC'	2	None
System ID	uomClass.PISystem.ID	22	Urlencoded Guid
UOM ID	uomClass.ID	22	Urlencoded Guid
Name Payload	`uomClass.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

2. ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'I'	1	None
Version	'1'	1	None
Marker	'UC'	2	None
System ID	uomClass.PISystem.ID	22	Urlencoded Guid
UOM ID	uomClass.ID	22	Urlencoded Guid

3. Path Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'P'	1	None
Version	'1'	1	None
Marker	'UC'	2	None
Name Payload	`uomClass.GetPath(AFEncodeType.Name, null).Substring(2).ToUpperInvariant()`	var	Urlencoded UTF8 String

4. Local ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'L'	1	None
Version	'1'	1	None
Marker	'UC'	2	None
UOM ID	uomClass.ID	22	Urlencoded Guid

5. Default ID Only Type

Field Name	Value	Encoded Width	Encoding Method
Type	'D'	1	None
Version	'1'	1	None
Marker	'UC'	2	None
UOM ID	uomClass.ID	22	Urlencoded Guid

Back to Table of Contents

WebID Encoding (WebID)

Supplied code snippets are using the C# programming language; the exact syntax may differ depending on the programming language used by your client application.

None

If an Encoding Method of *None* is specified, the verbatim value is represented directly in the WebID.

Urlencoded Bytes

If an Encoding Method of *Urlencoded Bytes* is specified, then a byte array input is expected, and the output is a URL-safe Base64 string, as specified in RFC 4648 Section 5. Importantly, WebID requires that padding not be used (see RFC 4648 section 3.2).

```
internal static string Encode(byte[] value)
{
    string encoded = System.Convert.ToBase64String(value);
    return encoded.TrimEnd(new char[] { '=' }).Replace('+', '-').Replace('
/', '_');
}
```

Urlencoded Guid

If an Encoding Method of *Urlencoded Guid* is specified, then the GUID is converted to a byte array using the CLR's `Guid.ToByteArray()` or similar, and then encoded using the method described in Urlencoded Bytes. When implementing on non-Microsoft platforms, note that the byte order for a GUID as produced by the CLR, or as defined in `Rpcdce.h`, is not the same as the byte order for an RFC 4122 UUID. WebID serialization and deserialization requires a Microsoft-style GUID byte array.

```
internal static string Encode(Guid value)
{
    byte[] bytes = value.ToByteArray();
    return Encode(bytes);
}
```

Urlencoded Int32

If an Encoding Method of *Urlencoded Int32* is specified, then the integer is converted to a little-endian byte array, and then encoded using the method described in Urlencoded Bytes.

```
internal static string Encode(int value)
{
    byte[] bytes = System.BitConverter.GetBytes(value);
    if (!BitConverter.IsLittleEndian)
    {
        Array.Reverse(bytes);
    }

    return Encode(bytes);
}
```

Urlencoded UTF8 String

If an Encoding Method of *Urlencoded UTF8 String* is specified, the String is converted to a UTF-8 byte array, and then encoded using the method described in Urlencoded Bytes.

```
internal static string Encode(string value)
{
    byte[] bytes = System.Text.Encoding.UTF8.GetBytes(value.ToUpperInvaria
nt());
    return Encode(bytes);
}
```

WebID Type (WebID)

Indicates the format in which WebIDs should be returned. The default type is configurable using the WebIDType configuration setting. The following values are accepted:

WebIdType	Description
Full	Encodes all information, similar to WebID version 1.0. Full WebIDs are longer, but are more resistant to items being moved, renamed, or deleted.
IDOnly	Encodes only object IDs into the WebID. IDOnly WebIDs are shorter, and will always refer to the same item, even if it is moved. However, IDOnly WebIDs will no longer be valid if the item is deleted.

WebIdType	Description
PathOnly	Encodes only path information into the WebID. PathOnly WebIDs will always refer to the same location in the AF hierarchy, regardless of which item is located there.
LocalIDOnly	This type is similar to IDOnly, but is dedicated to resources on the local (relative to the PI Web API instance) asset or data server. The local asset/data server ID and path information is left out. Note: This type is not persistent or unique, and could represent different items on different servers. **It is not compatible with load balancer configurations.**
DefaultIDOnly	This type is similar to IDOnly, but is dedicated to resources on the default asset server or data server. The default asset/data server ID and path information is left out. Note: This type is not persistent or unique, and could represent different items on different servers. **It is not compatible with load balancer configurations.**

Some of the WebID 2.0 formats can be generated in client application code rather than querying PI Web API for them.

AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com
To find your local AVEVA office, visit **www.aveva.com/offices**