Parature API Specification

Parature, Inc. I 13625 Suite B Dulles Technology Drive – Herndon, VA 20171 I Main (703) 564-7758 I Fax: (703) 564-7757

Contents

Parature API Specification

Introduction

Clients wishing to automate interaction with Parature Service Desk data can do so using a simple HTTP - based API. With the API, data can be added, updated, deleted and retrieved for the following object types: Customer, Account, Ticket, Product, Asset, KB and Download. The API also supports performing workflow actions on Ticket and Asset objects.

The API adheres to REST principles. See the following references for general information on REST-based APIs.

- http://en.wikipedia.org/wiki/REST
- http://www.peej.co.uk/articles/restfully-delicious.html
- http://www.onlamp.com/pub/a/python/2006/02/23/using-rest-with-ajax.html

Samples

Sample code using the API is available on the Parature Support Center.
(You must be logged into the Support Center to view this article.)
Samples are provided in Java, C#, Ruby and PHP here:

https://support.parature.com/link/portal/3/51/Article/3544/API-Code-Examples

Location

Live/Production

In the production environment, the API is accessed via your Service Desk host. For example, if you normally log in to the Service Desk at https://s3.parature.com/ics/service/login.asp, you will access the API at https://s3.parature.com/api/....

Development Sandbox

For development and testing, use the API against your sandbox data. Your sandbox contains a copy of your production data, allowing you to code against your real schema and data, without risk of damaging your live system.

The data in the sandbox is wiped and refreshed with a copy of your production data on the 1st and the 15th of every month. The data that is restored to the sandbox will be from a prior full backup, and not exactly from the 1st or 15th. If the 1st or 15th occurs on:

- Monday, Tuesday, or Wednesday - the data will be from a backup from the Monday.
- Thursday, Friday, Saturday or Sunday - the data will be from a backup taken on the Thursday

To access the sandbox via the API, use your service desk hostname, followed by "-sandbox". For example, if your production Service Desk host is s3.parature.com, your sandbox host will be s3-sandbox.parature.com. The sandbox contains a copy of your configuration and data and can be accessed using your normal credentials.
Please visit: https://support.parature.com/link/portal/3/51/Article/3490/About-Parature-Sandbox to find out more information about your sandbox environment and how to access it

Going Live With an Integration

To switch over from Sandbox to Live modify the API calls to use your production Service Desk hostname, and a token from one of your production CSRs or system accounts. It is good practice to parameterize these values.

API Throttling / Rate Limits

The APIs are throttled to levels which will prevent degrading the performance of your service desk and portal. It is advisable to include the ability to throttle your own code to prevent triggering more strict throttling rules, which engage when an API client is seen performing sustained, intense activity via the API. If a request is denied due to throttling, the system returns a 503 - Service Unavailable message.

API throttling is on a per-IP basis; thus, the system tracks requests coming from a particular machine. The throttling rate at which an IP address (aka client) is making requests is tracked regardless of the type of API request. If that rate exceeds 2 requests per second (120/minute), for a period of two minutes, the client is placed in 'throttled mode'. In 'throttled mode,' the client's request rate is reduced to one request every two seconds (30 per minute). The client remains in 'throttled mode' for 2 minutes (120 seconds) at which time they are returned to the normal throttling restrictions. At any point, if the current request rate is exceeded, the client receives a 503 error with a message "Too many requests, please try again later."

Some programming languages and frameworks, such as .NET, can limit concurrent socket connections. Make sure you properly close/dispose all request & response streams and handles in your code as suggested by the framework's appropriate documentation. In some cases, you may see behavior which may seem like Parature throttling in that the response doesn't come back immediately, but in fact is the library waiting to time-out an existing open socket handle before sending another request to Parature.

Generating an API Token

Access to the API is a two-step process. You must first have the API role. (Please contact your system administrator if you do not have this role.) Following this, you need to generate an API authentication token for the CSR or yourself, if you have been given the API role by an administrator.

If you have the API role, you can generate your own API Authentication Token:

1. Click on the **My Settings** button in the upper right hand corner of the Service Desk. Scroll down the page until you find the **API Authentication Token** section.
2. After confirming status of your token from the Token Status field, enter the number of days your authentication token should be active in the **Token Expiration (in days)** field. If you do not want your user's token to expire, leave the Token Expiration field blank.
3. Click the **Generate Token** button. A token will be generated on the screen.
4. You may use this token for any API operations that need to be performed.

It is also possible to delete the token. Deleting a token immediately removes any API access for operations performed using the deleted token.

1. Click on the My Settings button in the upper right hand corner of the Service Desk.
2. Scroll down the page until you find the **API Authentication Token** section.
3. Click on the **Delete Token** button.

As an administrator you can generate an API authentication token for any user with the API Role:

1. Click on the **Setup** tab.
2. Click on the Users and Role Management link.
3. Find the user whose you wish to give an API token to, and click on their name. The Edit User screen will appear in the right pane.
4. Scroll down the page until you find the **API Authentication Token** section.
5. After confirming status of the user's token from the **Token Status** field, enter the number of days the user will have an authentication token in the **Token Expiration (in days)** field. If you do not want your user's token to expire, leave the Token Expiration field blank.
6. Click the **Generate Token** button. A token will be generated on the screen.
7. This token may be used for any API operations that need to be performed.

It is also possible to delete the token. Deleting a token immediately removes any API access for operations performed using the deleted token.

1. Click on the **Setup** tab.
2. Click on the Users and Role Management link.
3. Find the user whose API token you want to delete, and click on their name. The Edit User screen will appear in the right pane.
4. Scroll down the page until you find the **API Authentication Token** section.
5. Click on the **Delete Token** button.

Best Practices

A set of recommendations and best practices when using the Parature API are listed below.

Schema Changes

The schema operation returns a "live" schema/object. This schema will change each time you add, delete or modify a custom field within Parature. For integrations that must automatically adjust to changes in the schema, the integration should read the schema prior to each run and diff it from the previous version to determine if there have been any changes.

Integrations and Parature Maintenance Windows

The Parature system is periodically taken offline for maintenance procedures. During these procedures you should stop your integration from running. If you cannot stop it, e.g. it is embedded inside larger site or application, you'll need to make the integration recognize and react to maintenance windows in an appropriate manner. When the Parature system is offline for maintenance, all API calls will be returned with a 502 HTTP code. See API XML Return Codes for a list of all return codes.

De-coupling an Integration

Some Parature customers embed API interactions directly into their own applications or web sites. Since it's likely that your system will be on a release schedule that does not align exactly with the Parature release schedule, it is a good idea to isolate interactions between the two systems to a isolated set of code. Then when changes occur to the Parature API, you need only change that section of code, and can avoid an internal change to your application. This design principle is called "de-coupling."

Security Considerations

Protect The Token

The Parature API is functionally equivalent to Service Desk access. The API makes the assumption that it is being used by trusted and privileged users, such as a CSR. As such, it does not enforce the same content filtering that the portal does. The portal assumes content is coming from less trusted, or non- trusted end users. For example, the API will allow you to set the content of a text field to any value you submit. Conversely, the portal will filter text field submissions for some types of disallowed content. If you will be submitting end-user-supplied content into the system via the API, you should be filtering it for malicious content such as JavaScript, otherwise you may open yourself to a Cross Site Scripting attack (XSS). See http://en.wikipedia.org/wiki/Cross-site_scripting for details on cross site scripting.

Filter Content

The Parature API is functionally equivalent to Service Desk access. The API makes the assumption that it is being used by trusted and privileged users, such as a CSR. As such, it does not enforce the same content filtering that the portal does. The portal assumes content is coming from less trusted, or non- trusted end users. For example, the API will allow you to set the content of a text field to any value you submit. Conversely, the portal will filter text field submissions for some types of disallowed content. If you will be submitting end-user-supplied content into the system via the API, you should be filtering it for malicious content such as JavaScript; otherwise you may open yourself to a Cross Site Scripting attack (XSS). See: http://en.wikipedia.org/wiki/Cross-site_scripting for details on cross site scripting.

Terminology

The following terminology will be used in this specification.

Object Type
A unique business entity type, such as the Ticket, Knowledge Base Article, Customer, etc... These correlate with Parature modules. The object type has a name and a set of fields that hold values pertinent to the type. For example, a "ticket" type contains fields such as "Assigned To" and "Date Created". All object types define a field called "ID" that holds a value uniquely identifying a specific object of that type.

Object
A single data record of a particular object type. All objects of a given type share the name set of fields, but each object has field values that are independent of other objects. Every object has an "ID" field value that is different from every other object of the same type.

Object Location
A URI that references the details of a particular object.

Field
A data item that has a name and a value. A field is defined as part of an object type and present on every object instance of the object type. The field value type could be a string, a date, a number, an option selected from a list, a Boolean (true/false), or the ID of an object.

Operation
A request that moves object data in and out of the Parature system. The Parature API supports six operations: Create, Update, Delete, Retrieve, Schema, and List.

Action
An action is an additional object transition that can take place when an object is updated or created; for example, assigning a Ticket to a CSR. Actions are only available for specific Objects: Ticket and Asset.

Supported Object Types

For each API-accessible Parature module, there is at least one defined object type. The following table defines the object-type value for every module.

Table: First-Order Object Types

Module	Object Type)_object-type(
Account	Account
Chat	Chat
Customer	Customer
CSR	Csr
Download	Download
Knowledge	Article
Product	Product
Product (Asset)	Asset
Ticket	Ticket

Table: Second-Order Object Types

The following objects are second-order objects that are used by the modules above. These objects are not first-order Parature business objects, however, they behave like business objects from the API perspective. Some of these objects are read-only (i.e., they cannot be created or modified through the API), such as Queues, CSRs, SLAs and Departments. Other objects, such as Csr, ProductFolder, ArticleFolder, and DownloadFoder, can be retrieved, created and updated through the API.

Related Module	Object Type)_object-type(
Account	Csr
Account	View
Customer	Csr
Customer	View
Ticket	Csr
Ticket	Department
Ticket	Status
Ticket	View
Product	ProductFolder
Knowledge	ArticleFolder
Download	DownloadFolder
Download	Eula (End User License Agreement)
Download	Sla (Permissions field)
Ticket	Queue
Account	Sla
Customer	Sla
Knowledge	Sla (Permissions field)
Customer	CustomerRole
Csr	Role
Csr	Status
Csr	Timezone

Operations

For each object that is available within the API, there are six operations that may be performed against these objects: **Create**, **Update**, **Delete**, **Retrieve**, **Schema**, and **List**. An additional operation, **Auth**, is available for the Customer object type. Each of the operations are detailed below. To perform an operation against an object, an HTTP method, such as GET, PUT, POST, etc., is invoked using a specially formatted URI. The format of the URI is as follows:

```
https://hostname/api/version/account-id/dept-id/object-type[/objectid|/schema|status|upload|view|/action/action-id|/status/status-id]?query-string
```

where the following definitions apply:

Element	Description	Example
hostname	hostname or IP address of API server	myfarm-sandbox.parature.com
api	API URI path element	api *(fixed value)*
version	API version	v1
account-id	client account ID	33
dept-id	client department ID	420
object-type	singular form of object type	Ticket , See First Order Objects and Second-Order Objects
object-id	the ID of a specific instance of an Object Type	123169
schema	indicates a schema request operation	/schema *(fixed value)*
status	indicates a status list operation	/status *(fixed value)*
upload	requests a file upload URI to post attachments	/upload *(fixed value)*
view	indicates an object view list operation *(Available only for some objects)*	/view *(fixed value)*
action/action- id	request schema for a specific action	/action/372
status/status- id	request status details for a specific status	/Ticket/status/987
query-string	object field parameters and API options	_token_=Y63UehX7fpJGKuOBR2OJvqmFq=2007-01-02
auth	request verification of credentials *(Available only for Customer)*	/Customer/auth *(fixed value)*

Using the example elements above, the full URI would be the following:

```
https://myfarm-sandbox.parature.com/api/v1/33/420/Ticket/123169?_token_=Y63UehX7fpJGKuOBR2OJvqmFq=2007-01-02=
```

There are a number of special URI parameters that provide additional information or control the behavior of the API. These are all surrounded with underscore characters like _this_. The underscore character will not be allowed in custom field names, so this prefix easily differentiates the Parature API-specific parameters from the client-specified parameters for list (query) operations.

Deleting Fields

To delete the value of a field use the Update operation, and leave the value for the field blank. This will direct the Update API to remove the field value.

Token

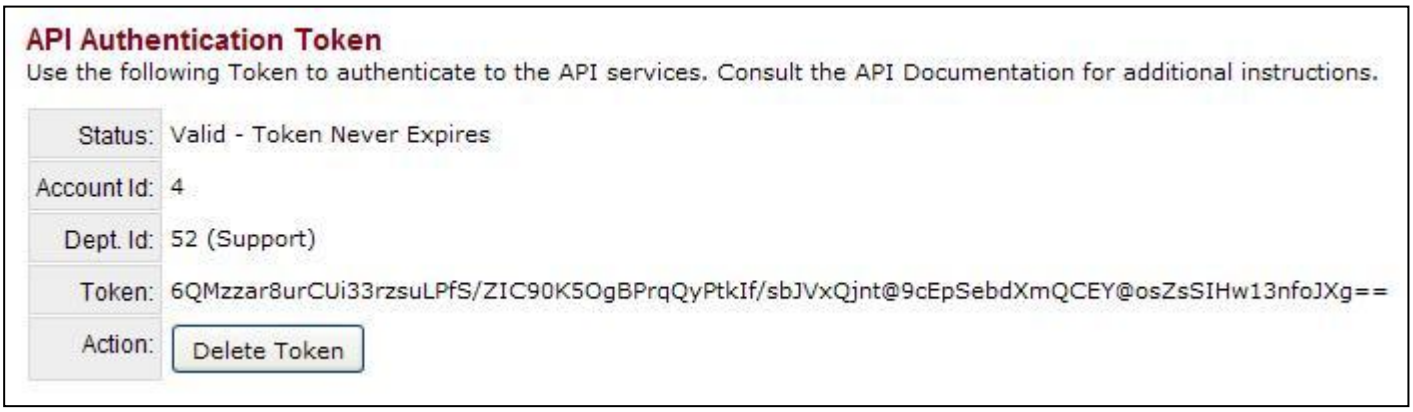
One of the special URI query-string parameters is the authentication token. It is used to verify the identity of the API user that is performing the operation against the Parature system. **The token is a required parameter for every operation.**

A CSR token is obtained by logging into the service desk and going to the "My Settings" page. Near the bottom of the page is the "API Authentication Token" section. If the CSR currently has a token generate d, the token will be shown in this section. If the CSR does not have a token generated, the "Generate Token" button is used to create a new one. Note that the "API Authentication Token" section is only available in the "My Settings" page if the CSR's account has an API license enabled and the CSR has an API Role granted.

If the password for the CSR's account is changed, the old token will no longer be valid (even if it is set to never expire) and a new token should be generated. To generate a new token go to My Settings, select to delete the old token, then click the button to generate a new token.

Obtaining Account and Department ID

The account and department ID are required on every API operation. This information can be obtained via the Service Desk by logging in as the user or CSR that has been granted the API Access role. Access the **My Settings** link in the top right menu of the Service Desk. Scroll down to the **API Authentication Token** section. The account and department ID values are shown as follows:



XML Object Formats

When performing the Create or Update operations the data is supplied in XML format within the HTTP POST or PUT request body. This XML data consists of an object representation of a single entity, e.g. Account, Ticket, etc. See Schema for format details when creating new objects.

Byte Order Marks

The XML data, which makes up the body of the HTTP response message, is prefixed with a Byte -order Mark (BOM). The BOM is a standard sequence of bytes that serve to indicate the "endianness" (big-endian or little-endian) for the type of encoding that is contained within the message. More information on the BOM can be found here. Since the API uses UTF-8 encoding, there is no byte-order to specify; it is used simply to indicate that the message encoding is UTF-8. The hexidecimal representation of the BOM for UTF-8 is EF BB BF.

Schema Files

The APIs include access to a set of schemas that will fully define the API XML format. The schemas are available in both XSD (W3C schema) and RNG (RelaxNG) formats at http://api.parature.com/schema/ It is good practice to validate XML input using these schema's prior to making and API call. The APIs will validate input against the schemas on the server side, and will return an error code 400 if the validation fails.

The example below shows a ticket entity XML submitted with an invalid field name tag.

```
<Ticket id="4" >
<Invalid_Field>This field doesnt exist on the Ticket object</Invalid_Field>
<Email_Notification>false</Email_Notification>
<Cc_Csr >emaile@company.com</Cc_Csr>
<Cc_Customer>FredFlintstone@parature.com,BarneyRubble@parature.com</Cc_Customer>
...
</Ticket>
```

When this is submitted to the API, for example as part of an update, the following error message is returned.

```
<?xml version="1.0" encoding="UTF-8"?><Error code="400" description="BadRequest"
message="Invalid XML data: The element 'Ticket' has invalid child element
'Invalid_Field'. List of possible elements expected: 'Assigned_To, Date_Created,
Date_Updated, Cc_Csr, Cc_Customer, Department, Email_Notification, Entered_By,
Ticket_Attachments, Ticket_Asset, Ticket_Customer, Ticket_Number, Ticket_Parent,
Ticket_Children, Ticket_Queue, Ticket_Status, Custom_Field, Action, ActionHistory'." />
```

XML Return Data Formats

All operations except Delete return XML data in response to the API request. See XML Return Code for a complete list of HTTP API response codes and their meanings.

On success, these operations return a 200 HTTP code, with XML data in one of three forms:

XML object

list of XML objects
This is the object/XML content returned from List calls.

XML Object Location
This is the XML content returned which points to the location of an XML object.

Parature API XML Object

This is the object/XML content returned from Retrieve calls.

The contents of a Parature XML object consists of an outer tag named after the object type, with nested tags representing entity fields, static fields, and custom fields. These tags are named according to their respective field names within Parature. Spaces are now allowed in XML tags, so any spaces within the field name are replaced with a single underscore character in the tag. The field value is represented as a text node with the field name tag. XML attributes are reserved for non-field information; typically these are values that should never need to be displayed to a user.

parature

Special Characters

There are certain special characters that may cause problems when submitting requests. This includes the use of special characters in ways that are illegal according to the XML Standard...

There are 5 such characters: < > " ' & If these are needed in a text string within the XML use escaping to replace them, per the following table.

Character	Entity
<	&lt;
>	&gt;
"	&quot;
'	&apos;
&	&amp;

The API assumes that you will escape any of these characters in the data that you submit into the system, therefore it also adds escaping to data it returns to you out of the system. It is possible that your system may contain data entered via the service desk that is already in an escaped form, particularly where HTML content is allowed, as in Ticket comments. If your data already contains escaped characters, the API will escape again. For example, if &amp appears in a text field, the API will detect the ampersand and return this as & Just un-escape it once as you receive it and re-escape it when you submit.

In addition, the set of non-printable ASCII control characters are not permitted according to the XML specification. None of these characters can be escaped and should be stripped from any data before sending it to the API. ASCII control characters may accompany data that is "copied and pasted" from a web browser or other applications, such as Microsoft Word - this is how the characters unexpectedly make their way into data sources. The range of character codes are as follows:

hexadecimal: x00-x08, x0B, x0C, x0E-x1F
decimal: 0-8, 11, 12, 14-31

Unique Identifier

Each XML Object contains a globally unique identifier. When storing information about an XML Object outside of the Parature system, this unique identifier should be used to reference the object. **Do not use the** id **attribute value of an Api XML Object, as it is not guaranteed to be globally unique.** The unique identifier, or uid is an XML attribute that accompanies every XML Object or reference to an XML Object. The following is an example of a reference to a Customer Object with the unique identifier:

```
<Ticket_Customer display-name="Customer" required="true" editable="true" data-
type="entity">
<Customer id="4" uid="4/52/Customer/4" href="https://my-
servicedesk.parature.com/api/v1/4/52/Customer/4">
<First_Name display-name="First Name">John</First_Name>
</Customer>
</Ticket_Customer>
```

Service Desk URI

Each entity contains an attribute, service-desk-uri, which indicates the URL that can be used to access the object on the Parature Service Desk. The attribute will show up on the main entity being retrieved, but not on the "reference fields" as shown in the example below:

```
<Ticket id="5" uid="4/52/Ticket/5" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Ticket/5" service-desk-
uri="https://myfarm-sandbox.parature.com/ics/tt/ticketDetail.asp?ticket_id=5">
...
<Assigned_To display-name="Assigned To" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">John Doe</Full_Name>
</Csr>
</Assigned_To>
...
</Ticket>
```

The following entities now have this attribute:

- Account
- Article
- Asset
- Customer
- Download
- Product
- Ticket

Note: The format of Service Desk URLs is subject to change with any release. Saving these URL's in another system is therefore not recommended. However, the URL returned by the API is compatible with the Service Desk at the time the call is made. This URL is the same as displayed for the entity when using the RSS output option.

Note (2): When browsing to a Service Desk URL, the user will be prompted to log in to the Service Desk unless they already have an active session.

Field Data Types

The XML Object that is returned from a Schema or Retrieve operation includes field metadata that describe the field, such as the data type, and access level (i.e., if the field can be modified). The following is an example of a static field that is returned within a Schema XML Object:

```
<Name display-name="Name" required="true" editable="true" data-type="string">
```

Custom fields are similar to static fields in format, except that custom fields all use the same tag name and contain a field identifier attribute. The following is an example of a custom field that is returned within a Retrieve XML Object:

```
<Custom_Field id="88285" display-name="Phone Number" required="false" editable="true"
data-type="string">703-555-8787</Custom_Field>
```

Both field types have a display-name attribute that contains the *Field Name* value that appears on the Parature Service Desk and Portal. The required and editable attributes contain Boolean values indicating whether or not the field can be modified and whether or not the field is required for create or update operations. The data-type field describes what type of data the field contains. The following is a list of field data Types:

Data Type	Description
string	Contains alphanumeric data
int	Contains numeric data
date	Contains date and time data expressed in ISO 8601 combined date/time format (Zulu time/GMT)
boolean	Contains the value true or false
float	Contains decimal data
entity	Contains a reference to a related XML Object
attachment	Contains a reference to a file in the Parature system than can be downloaded
option	Contains one or more selections from pre-defined list of options

Entity Data Type

Entity fields are references to a related API object, such as a CSR that is assigned to a Ticket. This relationship is represented in the Ticket object as a field with data-type="entity". Entities typically contain a subfield with an XML Object Location and additional data to further identify the entity. For example:

```
<Assigned_To display-name="Assigned To" required="true" editable="true" data-
type="entity">
<CSR id="26" uid="14/62/CSR/26"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/CSR/26">
<Full_Name display-name="Full Name">Jane Doe</Full_Name>
</CSR>
</Assigned_To>
```

The entity field above is **Assigned To** and its subfield is the **CSR** named Jane Doe. Using the XML Object location (i.e., the *href* value) for the CSR field and the API authentication token, a subsequent Retrieve operation can be performed to retrieve the CSR details.

Note that occasionally the schema will display what appear to be double tags in an entity field. A good example can be seen in the Customer object.

```
<Customer id="3813640" uid="4070/4440/Customer/3813640" href="https://myfarm-
sandbox.parature.com/api/v1/4070/4440/Customer/3813640">
<Email display-name="Email" required="true" editable="true" data-
type="string">mdarcy@seven.comVOID</Email>
<First_Name display-name="First Name" required="true" editable="true" data-
type="string">Mark</First_Name>
<Last_Name display-name="Last Name" required="true" editable="true" data-
type="string">D'Arcy</Last_Name>
<Account display-name="Account" required="true" editable="true" data- type="entity">
<Account id="167712" uid="4070/4440/Account/167712" href="https://myfarm-
sandbox.parature.com/api/v1/4070/4440/Account/167712">
<Name display-name="Name">FabrisLane</Name>
</Account>
</Account>
</Customer>
```

There appear to be two Account tags, one nested inside the other. One Account tag represents the Account field within the Customer object. The inner Account tag represents the entity to which the field points, which is an Account object. The tags represent two different things which happen to have the same name.

Boolean Data Type

The boolean data type accepts the values 'true' or 'false'.

Attachment Data Type

An attachment data type contains one or more Attachment references to a file within the Parature system. Each Attachment reference is a URI that can be used directly to download a copy of the file. Similar to an Object Location, the Attachment URI can be used with an HTTP GET method to retrieve the file. However, unlike the Object Location, the Attachment URI contains a valid, limited-time use, authentication token for the Parature File Management Service. The API authentication token is not used for this operation. The Attachment reference field also contains an XML attribute, filename, with the name of the file. The following is a sample Attachment data field:

```
<Attachments display-name="Attachments" required="false" editable="true" data-
type="attachment">
<Attachment guid="172abe4de93843debfa323976d8e1b91" filename="My_Attachment.jpg"

href="https://myfarm-
sandbox.parature.com/FileManagement/Download/172abe4de93843debfa32
3976d8e1b91?token=6cb2kFhMO6zxrE1..."/>
</Attachments>
```

Option Data Type

Option fields express selection list values, in which an end user chooses one or more values from a predetermined list. Selection lists are typically represented within a User Interface as drop-down lists, multiple checkboxes, or radio buttons. Option fields contain each of the selection values with an optional selected attribute indicating whether or not the option has been selected. For fields that are multi-option fields, i.e., more than one option can be selected at the same time, each of the options that are selected contain the selected attribute. Options that do not contain the selected attribute indicate that the option is not selected. Option fields also contain a multi-value attribute that indicate whether or not the field permits multiple selections. The following is an example of an option field:

```
<Custom_Field id="88277" display-name="Got Milk" required="false" editable="true"
data-type="option" multi-value="false">
<Option id="211">
<Value>Yes</Value>
</Option>
<Option id="212" selected="true">
<Value>No</Value>
</Option>
</Custom_Field>
```

parature

Field Dependencies

Parature can display or hide fields based on the option selected in another field. This feature is called field dependency. In our example there is an option field called Contact Method with three options (email, phone, IM). The system can be configured to display only the appropriate field for each option. The Email field is displayed when the Email option is selected. The Phone Number field is displayed when the Phone option is selected. The IM Name and IM Service fields are enabled when IM is selected. This is represented in the XML Object schema using an Enables tag within the Option.

Note that the Enables tag uses and XPath expression to indicate which field is enabled when that option is selected. For custom fields, the field ID is used in the XPath expression. In the example below the dependent fields have the following IDs:

Field	ID
Email	111222
Phone	111333
IM Name	111444
IM Service	111555

```
<Custom_Field id="90311" display-name="Contact Method" required="true"
editable="true" data-type="option" multi-value="false">
<Option id="181499">
<Value>Email</Value>
<Enables>/Ticket/Custom_Field[@id = 111222 ]</Enables>
</Option>
<Option id="181500">
<Value>Phone</Value>
<Enables>/Ticket/Custom_Field[@id = 111333 ]</Enables>
</Option>
<Option id="183139">
<Value>IM</Value>
<Enables>/Ticket/Custom_Field[@id = 111444 ]</Enables>
<Enables>/Ticket/Custom_Field[@id = 111555 ]</Enables>
</Option>
</Custom_Field>
```

If the field being enabled is itself an option field, Parature can be configured to display only a subset of options in that dependent option field. For example, let's assume we have a custom field called Operating System (id=111111). A second field called Version (id=222222) is dependent on it. When Windows is selected for Operating System, the version field only shows the options XP and 2003. When OSX is selected for Operating System, the version field only displays 10.0 and 10.5 as options. This configuration would be represented as shown in the example below.

Option field values are also identified by unique IDs. We'll assume the following values for the version field:

Option Value	ID
XP	100001
2003	100002
10.0	100003
10.5	100004

```
<Custom_Field id="111111" display-name="Operating System" required="true"
editable="true" data-type="option" multi-value="false">
<Option id="111122">
<Value>Windows</Value>
<Enables>/Ticket/Custom_Field[@id=222222]/Option[@id = 100001 or @id =
100002]</Enables>
</Option>
<Option id="111123">
<Value>OSX</Value>
<Enables>/Ticket/Custom_Field[@id=222222]/Option[@id = 100003 or @id =
100004]</Enables>
</Option>
</Custom_Field>
```

Notice that both the Windows and OSX option enable the same Version field (222222), but they each only enable the options that are relevant to the selection.

XML List of Objects

An XML list of objects consists of an outer Entities element, with an inner element for each object being returned. The Entities tag contains a number of attributes defining pagination for the returning data. The total attribute indicates the total number of entities that match the query. The results attribute indicates the number of results or objects being returned on the current page. The page attribute which page in the sequence has been returned (e.g. 2 for the second page of results). The page-size attribute indicates the number of results being returned on each page. The =href = attribute contains a URI identical to the one used to obtain the list, minus the token.

The entities enclosed within the Entities tag will depend on which object type was requested for the listing. In all cases enclosed entities follow a standard XML Object format for that object type, but with a limited number of fields returned. The returned Object XML does not include custom fields.

In the example below a listing of all Customers was requested (in this case, only two customers exist).

```
<Entities total="2" results="5" page="1" page-size="25" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Customer">
<Customer id="6" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Customer/6">
<Email display-name="Email" required="true" editable="false" data-
type="string">gkeenan@someplace.com</Email>
<First_Name display-name="First Name" required="true" editable="true" data-
type="string">Gareth</First_Name>
<Last_Name display-name="Last Name" required="true" editable="true" data-
type="string">Keenan</Last_Name>
<Account display-name="Find Account" required="true" editable="true" data-
type="entity">
<Account id="6" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Account/6">
<Account_Name display-name="Account Name" />
</Account>
</Account>
<Sla display-name="Service Level Agreement" required="true" editable="true"
data-type="entity">
<Sla id="10" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Sla/10">
<Name display-name="Name">Registered Enterprise Company</Name>
</Sla>
</Sla>
</Customer>
<Customer id="7" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Customer/7">
<Email display-name="Email" required="true" editable="false" data-
type="string">David.Brent@gmail.com</Email>
<First_Name display-name="First Name" required="true" editable="true" data-
type="string">David</First_Name>
<Last_Name display-name="Last Name" required="true" editable="true" data-
type="string">Brent</Last_Name>
<Account display-name="Find Account" required="true" editable="true" data-
type="entity">
<Account id="7" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Account/7">
<Account_Name display-name="Account Name" />
</Account>
</Account>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="7" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Sla/7">
<Name display-name="Name">System Default</Name>
</Sla>
</Sla>
</Customer>
</Entities>
```

Parature API XML Object Location

When an object, such as a Ticket or Product is successfully created or updated, the HTTP response body contains an XML Object Location for the newly created or updated object. The Object Location is a URI that references the details of a particular object. By combining this URI with the API authentication token (i.e., the _token_ parameter and its value) an object's details can be retrieved. The Object Location also includes a globally unique identifier that can be used to store information about the object in a data source outside of the Parature system. The following is a sample response from performing a Create operation to create a new Ticket or from updating the existing Ticket whose ID is 96:

```
<Ticket id="96" uid="14/62/Ticket/96" href="https://myfarm-
sandbox.parature.com/api/v1/14/62/Ticket/96" />
```

To get the details of this object, the HTTP GET method is used along with the Object Location and authentication token, as follows:

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/96?_token_=f4woGH3B...
```

An XML Object Location URI is also used in fields within an object that reference another object, called Entity Field

Operation Types

Schema Operation

This operation retrieves a description of an object type. The schema applies to the object XML passed as input and output, and includes descriptions of every field. Allowable option and action lists are also included. Because Parature is customized for every client there are client-specific differences in object field definitions; the schema is unique for every object type within a given client or department. However, the XML returned is not a true schema in that it does not follow a schema standard. Instead it more closely resembles an "empty"' object, with all fields defined, but no values set.

Standards-compliant schemas are available for use in client-side validation. However, these schemas tightly describe only the default or system fields within the Parature system. Custom fields, unique to your Parature configuration, are available described in the XML returned from the schema operation as defined above.

The protocol-compliant schemas are available in RelaxNG (rng) format. The RelaxNG schemas are available for each operation on each object type. To access the tickets browse to http://api.parature.com/schema/rng/Operation-Object.rng. So, for example, to access the Ticket schema for the Update operation browse to http://api.parature.com/schema/rng/Update-Ticket.rng Operation names are defined in Operations and the object types are defined in Supported Object Types

Usage

URI	.../account-id/dept-id/object-type/schema	
Method	GET	
Query	_token_=	authentication token
	output=	output format (json)
Body	Empty	
Returns	200 OK + Object Schema XML	Success
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Invalid permission
	500 Internal Server Error	Unrecoverable error within the Parature system

JSON Output

Schemas can optionally be returned in Javascript Object Notation (JSON) format, by using the =_output_=json option. See Parature API JSON for details.

Example

The following is a sample Schema operation request for a Customer object:
https://myfarm-sandbox.parature.com/api/v1/4/52/Customer/schema?_token_=A/xD04zEqDMMz9Jlq...

```
Output XML:
<Customer>
<Date_Created display-name="Date Created" required="true" editable="false" />
<Date_Updated display-name="Date Updated" required="true" editable="false" />
<Account display-name="Find Account" required="true" editable="true" data-
type="entity">
<Account>
<Account_Name display-name="Account Name" />
</Account></Account>
<Customer_Role display-name="Customer Role" required="false" editable="true" data-
type="entity">
<CustomerRole>
<Name display-name="Name"/>
</CustomerRole>
</Customer_Role>
<Email display-name="Email" required="true" editable="false" data-type="string" />
<First_Name display-name="First Name" required="true" editable="true" data-
type="string" />
<Last_Name display-name="Last Name" required="true" editable="true" data-
type="string" />
<Password display-name="Password" required="true" editable="true" data- type="string"
/>
<Password_Confirm display-name="Password (confirm)" required="true" editable="true"
data-type="string" />
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla>
<Name display-name="Name" />
</Sla>
</Sla>
<Status display-name="Status" required="true" editable="true" data-type="entity">
<Status>
<Name display-name="Name" />
</Status>
</Status>
<Custom_Field id="87862" display-name="Primary POC" required="false" editable="true"
data-type="boolean" />
<Custom_Field id="87863" display-name="Support Plan" required="true" editable="true"
data-type="option" multi-value="false">
<Option id="10">
<Value>Silver</Value>
</Option>
<Option id="11">
<Value>Gold</Value>
</Option>
<Option id="12">
<Value>Platinum</Value>
</Option>
</Custom_Field></Customer>
```

Retrieve Operation

This operation retrieves a single object. By default all object fields are returned in the output XML, except for the object action history. The _history_ parameter can be used to include the history for objects that support it.

Usage

URI	.../account-id/dept-id/object-type/object-id	
Method	GET	
Query	_token_=	authentication token
	output=	output format (json)
	history=true	include object history
Body	Empty	
Returns	200 OK + Object XML	Success
	400 Bad Request	Malformed URI or invalid qeury string parameters
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Insufficient permission
	404 Not Found	An object with =_object-id_ = does not exist
	500 Internal Server Error	Unrecoverable error within the Parature system

Example

The following is a sample Retrieve operation for an Account object: https://myfarm-sandbox.com/api/v1/4/52/Account/6?_token_=VKWHzhBGS... Output XML:

```
<Account id="6" uid="4/52/Account/6"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Account/6" creation-date="2007-
09-22T22:17:50.593" modification-date="2007-09-22T22:17:50.593">
<Account_Name display-name="Account Name" required="true" editable="true" data-
type="string">Calvertron</Account_Name>
<Owned_By display-name="Owned By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Fred Flintstone</Full_Name>
</Csr>
</Owned_By>
<Modified_By display-name="Modified By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
```

```
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Fred Flintstone</Full_Name>
</Csr>
</Modified_By>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="10" uid="4/52/Sla/10" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Sla/10">
<Name display-name="Name">Registered Enterprise Company</Name>
</Sla>
</Sla>
<Custom_Field id="87849" display-name="Partner Type" required="true"
editable="true" data-type="option" multi-value="false">
<Option id="58" selected="true">
<Value>OEM Partner</Value>
</Option>
<Option id="59">
<Value>Reseller</Value>
</Option>
</Custom_Field>
<Custom_Field id="87848" display-name="Certified Trainer" required="false"
editable="true" data-type="boolean">1</Custom_Field>
<Custom_Field id="87851" display-name="Partner Description" required="false"
editable="true" data-type="string">Calvertron manufactures and sells widgets in
additional to providing consulting service for their own line of widgets and our full
line of gadgets and products.</Custom_Field>
<Custom_Field id="87853" display-name="Certified Professionals" required="false"
editable="true" data-type="int">12</Custom_Field>
<Custom_Field id="87850" display-name="PLB Specialization" required="false"
editable="true" data-type="option" multi-value="true">
<Option id="7">
<Value>Product A</Value>
</Option>
<Option id="8">
<Value>Product B</Value>
</Option>
<Option id="9">
<Value>Product C</Value>
</Option>
<Option id="60" selected="true">
<Value>All</Value>
</Option>
</Custom_Field>
<Custom_Field id="87852" display-name="Enrollment Date" required="true"
editable="true" data-type="datetime">2007-06-06T00:00:00</Custom_Field>
</Account>
```

List Operation

This operation retrieves a summary list of objects for those objects that match the provided query. The list operation is the same for all modules except Knowledgebase and Ticket. The Knowledgebase module has the standard behavior defined below that applies to all object types, as well as a set of specialized search behaviors defined in the Knowledgebase Article API. The Ticket API provides some additional list behavior defined in Ticket List for My Tickets, Ticket List by Status Type, and the Ticket Account field.

A list operation returns multiple objects of a single module type within an entities tag. Each object that is returned in the list contains basic details of the object (useful when displaying the list) as well as the Object Location that references the full object details. See XML List of Objects for details.

parature

Usage

URI	.../account-id/dept-id/object-type?query-string	
Method	GET	
Query	_token_=	authentication token
	output=	output format (rss, json)
	startPage=	page number (starts at 1)
	pageSize=	number of results per page
	total=true=	return only the total object count; mutually exclusive with _startPage_ and =_pageSize_
	fields=	a comma-delimited list of custom field IDs to include in the resulting object list
	Field_Name=	field value match using field name, where *Field_Name* is the name of the field *(applies to static, non-custom, fields only)*
	FIDnnnn=	custom field value match using field ID, where *nnnn* is the ID of the custom field
	view=	ID of the Account, Customer, or Ticket view to use for the query
	_status_type_=	Ticket status type filter, either open or closed. *(For Ticket List only.)*
	_my_tickets_=true	return only Tickets that are currently assigned to the CSR identified by the accompanying token.*(For Ticket List only.)*
	order=	a comma-delimited list of field names to specify the result set sort order. *(See Advanced Query Operations for more information.)*

Body	Empty	
Returns	200 OK + Object List XML or Object Count XML	Success
	400 Bad Request	Malformed URI or invalid querystring parameters
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Insufficient permission
	404 Not Found	An object with =_object-id_ = does not exist
	500 Internal Server Error	Unrecoverable error within the Parature system

parature

List Query Parameters

The default behavior of the list operation returns all objects of that type. The list can be reduced to a smaller subset by specifying field-based matches. For example, the customer list operation will return only customers with a last name of 'Smith' by specifying Last_Name_=Smith on the request.

A query parameter value specifies an exact match unless a modifier is added. For example, specifying Last_Name_=Smith will only match customers that have the last name of Smith, but not those with the last name of Smithson. Non-exact queries can be performed for object fields of type string, date, int or entity. See Field Data Types for a complete listing of data types. In addition, some Advanced Query Operations can be performed, such as sorting the results by a given entity field, or providing a list of possible values for exact and non-exact matches. An example of the latter case is to query for a list of Tickets where the Assigned_To field is a list of known CSRs.

The query modifiers are appended to the field name parameter, and are always surrounded by underscore characters, as shown below.

Modifier	Description	Example	Applies To
like	substring match for string fields	Name_like_=Smith	string
min	minimum value for scalar and date fields	Date_Created_min_=2007-03-04	date, int
max	maximum value for scalar and date fields	Cost_max_=250.00	date, int
_id_	the value represents an ID for object fields	Product_id_=11228	entity

A list can be filtered by a specific field value by specifying either its name (Name=Fred) or, for custom fields, its unique ID (FID552=Fred). If the field is an option type, the value must be specified by its ID as well, e.g. FID1234=5465

When performing a query using a date field, the following two formats are accepted:

Format	Description	Example
MM/DD/YYYY	This searches by date only	Date_Created_min_=02/15/2008
YYYY-MM-DD	This searches by date only	Date_Created_min_=2007-11-01
YYYY-MM-DDThh:mm:ssZ	This searches by date and time. Note that the time must be in GMT. This is expressed by the "Z" character at the end of the time value (for Zulu time). Also note that the "T" character is required to indicate the time portion of the value.	Date_Updated=2007-11-01T15:30:00Z

Examples

List Example 1- Simple List

List all customers in department 420 of account 33

```
https://myfarm-sandbox.parature.com/api/v1/33/420/Customer?_token_=ABC123XYZ...
```

List Example 2- List by entity field value and creation date

List all tickets in department 420 of account 33, assigned to Sam Baker (id=16), and created on or after April 5, 2007. Start at the first result and return a maximum of 25 tickets.

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?Assigned_To_id_=2132&Creation_Da
te_min_=2007-04-05&_start_=0&_results_=25&_token_=ABC123XYZ...
```

List Example 3- List Count

List the total number of Tickets in department 62 of account 14

```
https://myfarm-
sandbox.parature.com/api/v1/14/62/Ticket?_total_=true&_token_=VKWHzhBGS... Output
XML:
<Entities total="13"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket?_total_=true"/>
```

List Example 4- List including custom fields

List the Tickets in department 62 of account 14, and include the Summary (id=90360) and Details (id=90361) fields in the result set

```
https://myfarm-
sandbox.parature.com/api/v1/14/62/Ticket?_fields_=90360,90361&_token_=VKWH
zhBGS...
```

RSS Feeds and The Output Option

You can obtain an RSS feed from a list by specifying the _output_=rss option on the request query string. The RSS feed URL can be inserted into any standard RSS feed reader. For example:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?_output_=rss&_token_=ABC123XYZ...
```

The resulting RSS feed will contain only the fields that are displayed by default on the list operation for that object type. To make the RSS feed more useful and readable you can include custom fields in the feed using the *fields* options. For example, to include a custom field called Summary (id = 88710) and field called description (id = 67584) you would format the request like:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?_output_=rss&_fields_=88710,67584&_token_=A
BC123XYZ...
```

The custom field ids shown above are just examples and are different for each Parature customer. To obtain the values for your configuration request a schema/template for the object type you are referencing.

Sorting Results

It is possible to specify sort order on results returned from a list operation using the *order* paratuer. The format to specify a field by which results are sorted is _order_=Field_Name_asc_ or _order_=Field_Name_desc_. The _asc_ or _desc_ field name modifiers are used to specify **Ascending** or **Descending** order. A maximum of five fields can be specified for sorting by providing a comma-separated list of fields. The order of the comma-separated list indicates the order of the field sorting. For example, to sort a result of Customers by Last_Name (ascending), then by First_Name (ascending), the URI would be:

```
.../api/v1/123/456/Customer?_order_=Last_Name_asc_,First_Name_asc_...
```

Sorting String and Text Fields

Sorting rules only apply to alphanumeric characters for string (text) fields. Characters such as & , . ( ) - ? are ignored. For example, given the strings, "Test 123", "Test123", and "Test-123", the order of the result set sorted in ascending order would be: "Test123", "Test-123", and "Test 123". Non-alphanumeric characters do not follow the ASCII character order.

Sorting Entity Fields

When sorting by an entity field, such as Ticket_Customer (i.e., the Customer object associated with a Ticket), the entity's reference field is used to impose the order. A "reference" field is the field that names or identifies a given instance of an entity. For example:

```
<Ticket id="5" uid="4/52/Ticket/5" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Ticket/5" service-desk- uri="https://myfarm-
sandbox.parature.com/ics/tt/ticketDetail.asp?ticketID=5">
...
<Assigned_To display-name="Assigned To" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">John Doe</Full_Name>
</Csr>
</Assigned_To>
...
</Ticket>
```

In the above example, the reference field for the Assigned_To entity field is the Full_Name field of the Csr, John Doe. The associated URI for sorting Tickets by the Assigned_To field would be:

```
.../api/v1/123/456/Ticket?_order_=Assigned_To_asc_...
```

This would sort the Tickets by the CSR's Full_Name field that is assigned to the Ticket.

Query Using a Set of Field Values to Match ("IN" Search Criteria)

In addition to filtering a List of entities by a single value, you can specify a set of values, separated by commas. When more than one value is specified, the Parature API interprets the query search criteria as Field = value1 OR Field = value2 OR Field = value3... For example, to obtain a list of Customers, whose last name is either Smith or Jackson, the request URI would be:

```
.../api/v1/123/456/Customer?Last_Name=Smith,Jackson
```

This "OR" ability is only limited to values within a SINGLE field. This means that a request like "Last name is Smith OR First name is Joe" cannot be performed, as this is a cross field "OR". (Last_Name and First_Name are two different fields.) Filters across fields operate with an 'AND' behavior. For example, specifying Last_Name=Smith along with First_Name=Joe, would return only those Customers named Joe Smith.

The ability to perform "OR" filtering is supported for all field types except for boolean.

Including a Comma in the Search Criteria

If a comma character is to be included in the "OR" value search criteria, the comma must be "escaped" by preceding the comma with a slash character. For example, given a custom field named "Formal Name" (ID: 1001) in the Customer module, with all values stored in a format of Last Name, First Name, such as "Smith, Joe", the field value search criteria would be specified as follows:

```
.../api/v1/123/456/Customer?FID1001=Smith\,%20Joe&_toke_=...
```

The List request above would match all Customers with a Formal Name equal to "Smith, Joe".

Sample Scenario - Customer

Given the following list of Customers:
Customer 1
First Name: Joe
Last Name: Smith
Customer 2
First Name: Tom
Last Name: Washington
Customer 3
First Name: Joel
Last Name: Washington

Exact Matches (the "IN" value list)

To get a list of Customers whose last name is Smith, the URI would be:

```
.../api/v1/123/456/Customer?Last_Name=Smith&_token_=...
Customer 1 is returned in the result.
```

To get a list of Customers whose last name is Washi, the URI would be

```
.../api/v1/123/456/Customer?Last_Name=Smith,Washi&_token_=...
```

Customer 1 and Customer 3 are returned in the result. Although Customer 2's last name contains the word "Washi", it is only a partial match. Since this is an exact match, partially matched results are not returned.

Non-Exact Matches (the "IN + LIKE" value list)

To get a list of Customers whose last names are **like** "SMI", the URL would be:

```
.../api/v1/123/456/Customer?Last_Name_like_=Smi&_token_=...
Customer 1 is returned in the result.
```

To get a list of customers with a Last_Name **like** "Smi" or "Washi", the URI would be:

```
.../api/v1/123/456/Customer?Last_Name_like_=Smi,Washi&_token_=...
```

Since this is a "LIKE" match, all 3 Customer records are returned.

Relative Date Queries

A specific date range can be specified as search criteria by using the _min and _max_ field modifiers for date fields. This general date range functionality can be used by API clients for any date related query. However, using _min and _max to perform relative date queries, i.e., date criteria relative to the current date and time, the burden of dynamically calculating the date criteria is on the API client. As a convenience feature, the API provides a method for specifying relative date and time values as query parameters, for example, a list of Tickets created yesterday, last week, or within the past 10 days. Relative dates are expressed on the request URI using a set of reserved words, such as _yesterday_or _last_week_ in the following manner:

```
http://my-
servicedesk.parature.com/v1/123/456/Customer?Date_Created=_last_5_days_&_token_=Gv7@b
... http://my-
servicedesk.parature.com/v1/123/456/Customer?Date_Updated=_today_,_yesterday_&_token_
=Gv7
@b...
```

Relative Date Keywords

The following table describes the URI query string keywords for specifying relative dates:

Keyword	Definition
today	12:00:00.000am to 11:59:59.999pm **on the date** the operation is performed.
yesterday	12:00:00.000am to 11:59:59.999pm on the date **before** the operation is performed.
tomorrow	12:00:00.000am to 11:59:59.999pm on the date **after** the operation is performed.
_last_n_days_	12:00:00.000am *n* days preceding the day operation is performed through 11:59:59.999pm on the day the operation is performed. (In practice, this results in "n+1" days worth of results.)
_next_n_days_	12:00:00.000am on the day the operation is performed through 11:59:59.999pm *n* days following the day the operation is performed. (In practice, this results in n+1 days worth of results.)
_this_week_	12:00:00.000am the Sunday immediately prior to the day on which the operation is performed through 11:59:59.999pm on the Saturday immediately following the day on which the operation is performed. If the operation is performed on a Sunday the date range will begin on the day the operation is performed. If the operation is performed on a Saturday the date range will end on the day the operation is performed.
_last_week_	12:00:00.000am the next to last Sunday prior to the day on which the operation is performed through 11:59:59.999pm on the Saturday immediately preceding the day on which the operation is performed. If the operation is performed on a Sunday the date range will begin on the prior Sunday. If the operation is performed on a Saturday the date range will end on the prior Saturday.
_next_week_	12:00:00.000am the next to last Sunday following the day on which the operation is performed through 11:59:59.999pm on the Saturday immediately following the day on which the operation is performed. If the operation is performed on a Sunday the date range will begin on the prior Sunday. If the operation is performed on a Saturday the date range will end on the prior Saturday.
_this_month_	From 12:00:00.000am on the first day of the month through 11:59:59.999pm on the last day of the month in which the operation is performed.
_last_month_	From 12:00:00.000am on the first day of the month through 11:59:59.999pm on the last day of the month immediately prior to the month in which the operation is performed.

_next_month_	From 12:00:00.000am on the first day of the month through 11:59:59.999pm on the last day of the month immediately following to the year in which the operation is performed
_last_n_months_	12:00:00.000am on the first day of the month *n* months preceding the month in which the operation is performed through 11:59:59.999pm on the last day of the month immediately preceding the month in which the operation is performed. (This results in *n* months worth of results.)
_next_n_months_	12:00:00.000am on the first day of the month immediately following the month in which the operation is performed through 11:59:59.999pm on the last day of the month *n* months following the month in which the operation is performed. (This results in *n* months worth of results.)
_this_year_	From 12:00:00.000am on the first day of the year through 11:59:59.999pm on the last day of the year in which the operation is performed.
_last_year_	From 12:00:00.000am on the first day of the year through 11:59:59.999pm on the last day of the year immediately prior to the year in which the operation is performed.
_next_year_	From 12:00:00.000am on the first day of the year through 11:59:59.999pm on the last day of the year immediately following to the year in which the operation is performed.

Specifying Time Zones

When using relative date keywords, the date-time reference for when the operation is performed is assumed to be UTC/GMT. However, API clients may want to specify a different time zone as the reference. To do so, a time zone suffix is added to the keyword. The suffix indicates the number of hours to add to or subtract from the base UTC time. The syntax of the format is as follows:

```
_ keyword _UTC_<plus|minus>_n_
```

where *keyword* is one of the relative date keywords above, plus or minus indicate addition or subtraction and n is the number of hours to be added or subtracted. For example: _last_5_days_UTC_minus_5_ indicates 12:00:00.000am 5 days ago through 11:59:59.999pm today, where today is defined as the current date-time GMT minus 5 hours, or, EST.

Listing by Views

Within the Service Desk, **Views** can be created for Accounts, Customers, or Tickets. A **View** is a specialized query in which users can specify the search criteria and the fields to be returned in the result set. For example, a view can be created to return all Customers with a status of **Pending**. The API List operation can operate against these views to return results as specified in the search criteria. The fields that are returned in the API List operation, differ slightly from what is returned in the Service Desk. Within the Service Desk, only the fields that make up the view are shown in the results. **In the results of an API view list, all static fields are returned in the result set along with all custom fields that are defined in the view**.

Views definitions cannot be created or modified via the API. They can only be created and modified via the Service Desk. To perform an API List operation using a view, the view id is used to specify which view to apply to the operation. A list of available views can be obtained by performing a List operation on the entity (Account, Customer, or Ticket) for which the view was created. The view id is then specified as a URI query-string parameter when the entity List operation is performed.

Listing Available Views

The format for specifying a list of available views for a particular entity (object-type) is

```
https://host/api/version/account-id/dept-id/object-type/view?query-string
```

where object-type is either the Account, Customer, or Ticket object type. For example, the following URI requests a list of all Ticket views:

```
https://my-
servicedesk.parature.com/api/v1/123/456/Ticket/view?_token_=I7GNr0G7ngMz...
```

The following is sample output of the View List operation:

```
<Entities total="3" results="3" page="1" page-size="25"
href="https://my-servicedesk.parature.com/api/v1/4/52/Ticket/view">
<View id="5" uid="4/52/Ticket/view/5" href="https://my-
servicedesk.parature.com/api/v1/4/52/Ticket/view/5">
<Name display-name="Name" required="false" editable="false" data-
type="string">Test Ticket View</Name>
</View>
<View id="6" uid="4/52/Ticket/view/6" href="https://my-
servicedesk.parature.com/api/v1/4/52/Ticket/view/6">
<Name display-name="Name" required="false" editable="false" data-
type="string">Another Ticket View</Name>
</View>
<View id="7" uid="4/52/Ticket/view/7" href="https://my-
servicedesk.parature.com/api/v1/4/52/Ticket/view/7">
<Name display-name="Name" required="false" editable="false" data- type="string">My
View</Name>
</View>
</Entities>
```

Query By View

To perform a List operation using a view, the View id value is specified as a query-string parameter along with the parameter name _view_. For example:

```
https://my-
servicedesk.parature.com/api/v1/123/456/Ticket?_view_=7&_token_=I7GNr0G7ngMz...
```

Would return the contents of view 7, which from the list above is called "My View".

A query by view can be combined with the some of the API's advanced query operations to perform additional operations on the result set, for example limiting view list of Customers to only those with Last_Name field **like** "Smi", sorted in ascending order. The URI for this request would be the following:

```
.../api/v1/123/456/Customer?_view_=789&Last_Name_like_=Smi&_order_=Last_Name_asc_&_
token_=I7GNr0G7ngMz...
```

The query will return the normal contents of the view, further reduced to include only entities with a Last_Name like "Smi".

Note that any field-based advanced query operations must operate on the fields already returned by the view. For example, the query above would only work if Last_Name were one of the fields returned by the view.

Results

The list returned from the query be view includes all static field values on the entity and all custom fields displayed in the view on the service desk.

Create Operation

This operation creates a new object. All required object fields must be specified in the input XML.

A list of field descriptions for the object can be obtained by using the Schema operation. The object schema describes the metadata associated with each field; for example, the data type and whether or not the field is required. If an Action is executed upon creating an object, it is advised to retrieve the object immediately after creating the object, since an action often changes the property of one or more fields of the object.

Usage

URI	…/account-id/dept-id/object-type	
Method	POST	
Query	_token_=	authentication token
	enforceRequiredFields	If **false**, ignores dependency and required field validation for custom fields. Default = true. See Bypassing Validation for details
Body	Object XML	
Returns	201 Created + Object Location XML	Success
	400 Bad Request	Malformed URI or malformed Object XML
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Insufficient permission
	500 Internal Server Error	Unrecoverable error within the Parature system

Example

The following is a sample Create operation for a Customer object: URI =

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Customer?_token_=VKWHzhBGS...
```

Input XML:

```
<Customer>
<Email>jsomeone@parature.com</Email>
<First_Name>Jane</First_Name>
<Last_Name>Someone</Last_Name>
<Account>
<Account id="27"/>
</Account>
<Sla>
<Sla id="17"/>
</Sla>
<Custom_Field id="123">
<Option id="565" selected="true" />
</Custom_Field>
</Customer>
```

Returns: On success returns a 201, with response body consisting of the ID and URI location of the newly created customer object, Output XML:

```
<Customer id="42" uid="14/62/Customer/42" href="https://myfarm-
sandbox.parature.com/api/v1/14/62/Customer/42" service-desk-
uri="https://myfarm-sandbox.parature.com/ics/customer/custDetail.asp?customerID=42"
/>
```

Default Values for Custom Fields

Custom fields that are configured within the Service Desk can contain default values. The default values are applied during the Create operation when a value is not supplied in the XML input object. Default values are applied for both "required" and "optional" fields. When field dependencies exist, dependent field default values are not applied. In other words, if a default value activates a dependent field, and the dependent field has a default value, the dependent field's default value is not applied.

Stripping of White Space

When a static field with a data type of string is submitted for a Create or Update operation, the API removes any extraneous white space. For details on how white space is removed, see the description of the Update operation.

Validation Behavior

See Validation Behavior for details.

Update Operation

This operation updates an existing object. The operation behavior is identical for all entity types except Customer, which follows this behavior plus an additional behavior documented at Notifying Customer of Password Change

Usage

URI	.../account-id/dept-id/object-type/object-id	
Method	PUT	
Query	_token_=	authentication token
	enforceRequiredFields	If false, ignores dependency and required field validation for custom fields. Default = true. See Bypassing Validation for details
Body	Object XML	
Returns	201 Created + Object Location XML	Success
	400 Bad Request	Malformed URI, Object XML or invalid querystring parameters
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Insufficient permission
	404 Not Found	An object with =_object-id_ = does not exist
	500 Internal Server Error	Unrecoverable error within the Parature system

The following is a sample Update operation for an Account object: https://myfarm-sandbox.parature.com/api/v1/14/62/Account/12?_token_=VKWHzhBGS... Input XML:

```
<Customer id="9">
<Email>dtm@place.com</Email>
<First_Name>Dawn</First_Name>
<Last_Name>Tinsley</Last_Name>
<Account>
<Account id="6" />
</Account>
<Sla>
<Sla id="10" />
</Sla>
<Custom_Field id="87864">
<Option id="15" selected="true" />
</Custom_Field>
<Custom_Field id="87863">
<Option id="10" selected="true" />
</Custom_Field>
<Custom_Field id="87862">0</Custom_Field>
</Customer>
```

Output XML:

```
<Account id="12" uid="14/62/Account/12" href="https://myfarm-
sandbox.parature.com/api/v1/14/62/Account/12" service-desk-
uri="https://myfarm-sandbox.parature.com/ics/am/amDetail.asp?amID=12" />
```

For a more detailed example of updating an object see the account example.

Validation Behavior

See Validation Behavior for details

Read then Update

To perform an update, it is easiest to follow these steps:

1. Retrieve the details of the object to be updated
2. Apply any changes to your local XML: add any new fields, replace the value of existing fields, delete existing fields, change dropdown selections, et cetera.
3. Strip all attributes from the tags.
4. Make the update call using the modified XML.

This ensures the object is updated, with no loss of data other than intentional removals.

Stripping of White Space

When a static field with a data type of string is submitted for a Create or Update operation, the API removes any extraneous whitespace. Whitespace is defined as the Carriage Return character (x0d), the Line Feed character (x0a), or the Tab character (x09). At most, a single space character (x20) is preserved between substrings. The rules by which the API strips white space for static fields is as follows *(white space characters are represented as follows [CR]=Carriage Return, [LF]=Line Feed, [TAB]=Tab)*:

1. When a field contains leading or trailing whitespace, the whitespace is removed. **In this case, the Space character is considered white space.** For example, given the field value " hello world ", the result is "hello world".
2. When a field contains two strings that are separated one or more Space and/or Tab characters, a single
Space character replaces the extraneous characters. For example, given the field value "hello [TAB] world", the result is "hello world".
3. When a field contains two strings that are separated only by a Carriage Retun and/or Line Feed, the Carriage Return and/or Line Feed are removed. For example, given the field value "hello[CR][LF]world", the result is "helloworld".
4. When a field contains two strings that are separated by a Carriage Retun and/or Line Feed and a Space character, the Carriage Return and/or Line Feed are removed, but the Space is preserved. For example, given the field value "hello [CR][LF]world", the result is "hello world".
5. Some static fields must allow Carriage Return, Line Feed, and Tab characters, such as the Question and
Answer fields for the Article object. The API does not strip white space for any of these fields.
6. **The API does not strip extraneous whitespace for Custom Fields.**

The following table shows which XML Object string fields that are affected by whitespace and whether or not the whitespace is stripped:

Object Type	Field	Whitespace Removed
Account	Account_Name	Yes
Article	Question	No
Article	Answer	No
ArticleFolder	Name	Yes
ArticleFolder	Description	No
Asset	Name	Yes
Asset	Serial_Number	Yes
Customer	Email	Yes
Customer	Password	Yes
Customer	Password_Confirm	Yes
Customer	First_Name	Yes
Customer	Last_Name	Yes
Customer	User_Name	Yes
Download	Name	Yes
Download	Description	No
Download	Title	Yes
Download	Guid	Yes
Download	External_Link	Yes
DownloadFolder	Name	Yes
DownloadFolder	Description	No
Product	Name	Yes
Product	Sku	Yes
Product	Price	Yes
Product	Shortdesc	Yes
Product	Longdesc	No
ProductFolder	Name	Yes
ProductFolder	Description	No
Ticket	Cc_Csr	Yes
Ticket	Cc_Customer	Yes

Deleting a Field Value

To delete a value, perform an update on the object, including all required parameters. For the field to be deleted, include the field's node/tags, but do not include a value. The API will remove the existing value for that field. So, for example, if a Customer has a First Name value of Fred, and you submit an update with a blank First Name field, the name Fred will be removed the field will be blank, e.g.

```
<First_Name></First_Name>
```

The exception to the deletion are **non-editable fields**, which are marked with the attribute

```
editable="false"
```

Non-editable fields cannot be deleted, and the API will prevent any attempt to do so.

This behavior is consistent for option/selection fields as well. If the outer field tags are included in the update, but no options, all options will be de-selected.

If the tags are completely absent from the request, the field value will remain unchanged.

Actions

In addition to standard updates the API supports the execution of actions against certain object types. Actions are object transitions defined within the workflow of the object. For example, most ticket workflows support the concept of a close action, which a Customer Service Representative will use to complete a ticket. See the Ticket API page for details on performing actions on tickets.

The set of allowable Actions for an object varies according to its current state (its field values), so it is best to first retrieve the object before updating it in order to get the list of possible Actions. Also, an Action often changes the property of one or more fields of the target object, so it is advised to retrieve the object immediately after updating the object.

Delete Operation

This operation deletes an existing object. For some Parature objects/modules on the Service Desk, deletion is a two-step process. First the object is trashed, which moves the object to a trash bin. Then the object can be purged, which permanently deletes it from the system. The rationale for the two step delete process is as a preventative against data loss on key business entities by allowing the user to recover objects from the trash if the deletion was in error. A trashing process is also available on the Service Desk for these objects. For other objects, deletion is single-step and always permanent.

Objects which support the two-step, "delete then trash," style of deletion are: Account, Customer, KB Article, Download, Ticket, Asset, and Product. All other objects use a simple, single-step delete.

These alternate methods are accomplished through the API using the _purge_ paramter as outlined below.

Usage

URI	.../account-id/dept-id/object-type/object-id	
Method	DELETE	
Query	_token_=	authentication token
	purge=true	Object is permanently deleted. Default if left blank = false
Body	Empty	
Returns	204 No Content	Success
	400 Bad Request	Malformed URI or invalid qeury string parameters
	401 Unauthorized	Invalid authentication token
	403 Forbidden	Insufficient permission
	404 Not Found	An object with =_object-id_ = does not exist
	500 Internal Server Error	Unrecoverable error within the Parature system

Two-Step Delete

Use on Account, Customer, KB Article, Download, Ticket, Asset, and Product *only*.

1. To perform a delete on one of these objects, first call the delete command with the option _purge_=false. This will cause the object to be moved into the "trashed" status. An object can be left in this status for a period of time, or you can delete it immediately.
2. To permanently delete an object that is in the trashed status, call the delete command on the object again with the option _purge_=true. **Use with caution. When using the _purge_=true the object will be permanently deleted from the system and cannot be recovered.**

Note: to perform the purge you must already know the ID of the object to be purged, as trashed objects do not show up in list queries.

Single-Step Delete

To delete and object that does not support trashing, simply call the delete command with the option _purge_=true. **Use with caution, as the object will be permanently deleted from the system and cannot be recovered.** If you call the delete command with the option _purge_=false the system will return a 400 error, as you are asking it to trash an object that doesn't support trashing.

Sample

The following is a sample Delete operation URL for a Ticket object. The URL would be submitted with an HTTP DELETE method.

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/8675309?_token_=VKWHzhBGS...
```

Parature field dependency system involves Parent fields and Child fields. Option fields are the only fields capable of functioning as Parents. A given Child field is activated (becomes visible on the portal and service desk) when a particular value is chosen on the Parent field. See the Parature User Guide section ‘Create Field Dependencies’ for details on how to configure this behavior in Parature.

When validation is enforced on API calls the following behaviors are exhibited when setting values for custom fields:

API Create/Update Request Tries to...	Resulting Action
Set Parent value that activates Child and valid Child field value is provided	Update the Parent and Child with values provided
Set Parent value that activates Child and a non-valid Child value is provided	Validation error (400/Bad Request)
Set Parent value that deactivates existing Child, activates a different Child, and new valid Child value is provided	Update the Parent and Child with values provided. Old Child field value is deleted.
Set Parent value that deactivates existing child, activates a new Child, and NO Child is provided	If Child field is a required field=API call returns validation error (400/Bad Request). Leaves existing Parent and Child values. If new Child field is not a required field=Updates the Parent value, and deletes the old Child value.
Parent deactivates existing child and invalid new value is provided	Throws validation error (400/Bad Request) and leaves existing
Parent field value is deselected (e.g. no selection) and has an existing Child value.	Parent value deselected. Child value is deleted.

Bypassing Validation

- Motivation
- Request Parameter
- Example
- Warning

Motivation

Parature allows you to make changes to your field setup at any time. Those changes affect the schema of the objects as they are presented via the API. Some of those changes, such as required fields and field dependencies, could cause an integration to stop working. Since there are many mission critical integrations that that need to create/update objects in the Parature system, it was decided to add the ability to bypass the field validation system on API calls so you can make integrations more tolerant to changes in the field system. A request parameter is now available that will instruct the API to ignore validation of:

- Required custom fields (required system fields are still necessary).
- Field dependency checks

Request Parameter

When performing a Create or Update operation add the optional parameter "_enforceRequiredFields_" and give it a Boolean value, which has the following meanings:

- **true** - Performs all validation steps without change to previous API functionality.
- **false**- Bypasses validation of dependency activation and required values (custom fields only).

The default value (if parameter is not provided) is **true**. No change to the posted XML is required. NOTE: This URL parameter pertains only to the current API request being made. To disable validation for all requests the parameter must be set to false on all requests.

Example

Below is an example of using the URI in a Create operation to tell the API to bypass validation:

```
https://my-sandbox.parature.com/api/v1/2345/5342/Ticket?_enforceRequiredFields_=
false&_token_=abc
```

Warning

By using the parameter, you may be entering data into Parature Customer Service which may be missing required fields. If a CSR tries to edit the record in the Service Desk and it is missing values for required fields, the CSR will be forced to provide the missing values before saving.

The behavior exists today. You can choose to turn an existing field (that may have missing values) into a required field, and the application will not force you to provide values for all the existing records. Instead it requires you to provide the value the next time the record is saved. We are calling out this behavior as it may become more prevalent with the new option to bypass field validation.

Module-Specific API Details

Certain modules have extended behavior within the API. For example, the API supports performing actions against the Ticket module. Some modules have extra dimensions of data, such as action histories, or configuration based behaviors. These details are called out in the module-specific documentation below.

- CSR API
- Chat Module API
- Customer Module API
- Product & Asset Module API
- Ticket Module API
- Download Module API
- Knowledge Base Article API
- Account, Customer, and Ticket Views

Examples

- See this account example, which includes usage of the List, Details and Update operations, as well as handling entity fields and custom fields.

A set of sample programs are available in the Parature support portal. Search "API Code Examples" to find the article. This article includes links to download the source code of the samples.

Parature API CSR Object

First, read the API Operations section to understand how the API operations work for all objects. This page describes only those portions of the CSR API that are specific to the CSR object and assumes you are familiar with the default behavior.

Creating and Updating a CSR with Roles

A CSR may be created with any number of roles. If a CSR is created or updated without specifying roles, the CSR will have no Service Desk permissions and will be unable to login. As with other entities, if a CSR is being updated, all current roles applying to that CSR must be provided in the Update body if the CSR is to retain the same permissions. To add a role to a CSR, resubmit the CSR to the update, with the additional role(s) added to the list. See the Read then Update method of using the Update API.

To assign a role to a CSR you need to know the ID of the role. If you don't know the IDs of the roles you wish to assign, perform a CSR Role list by using the following:

https://my-sandbox.parature.com/api/v1/415/688/Csr/role?_token_=VKWHzhBGS...

You'll get a list of roles back that will look similar to the following:

```
<Entities total="42" results="25" page="1" page-size="25" href="https://my-
sandbox.parature.com/api/v1/415/688/Csr/role">
...
<CsrRole id="227" uid="415/688/Csr/role/227" href="https://mysandbox.
parature.com/api/v1/415/688/Csr/role/227">
<Description display-name="Description" required="false" editable=" true" data-
type="string">Author</Description>
<Name display-name="Name" required="true" editable="true" datatype="
string">KM_Author</Name>
</CsrRole>
<CsrRole id="157" uid="415/688/Csr/role/157" href="https://mysandbox.
parature.com/api/v1/415/688/Csr/role/157">
<Description display-name="Description" required="false" editable=" true" data-
type="string">Chat Supervisor</Description>
<Name display-name="Name" required="true" editable="true" data-
type="string">Chat_Admin</Name>
</CsrRole>
<CsrRole id="158" uid="415/688/Csr/role/158" href="https://mysandbox.
parature.com/api/v1/415/688/Csr/role/158">
<Description display-name="Description" required="false" editable="
true" data-type="string">Chat Tech</Description>
<Name display-name="Name" required="true" editable="true" datatype="
string">Chat_Tech</Name>
</CsrRole>
...
</Entities>
```

See below for the XML content for a sample Create request. The CSR is assigned two roles.

```
<Csr>
<Email>foo@foo.com</Email>
<Full_Name>Foo Bar</Full_Name>
<Phone_1>555-555-2525</Phone_1>
<Phone_2>555-543-9876</Phone_2>
<Fax>555-234-5678</Fax>
<Role>
<CsrRole id="227"/>
<CsrRole id="158"/>
</Role>
<Screen_Name>fooman</Screen_Name>
<Status>
<Status id="-1"/>
</Status>
<Timezone>
<Timezone id="1"/>
</Timezone>
<Password>bar</Password>
<Date_Format>yyyy/mm/dd</Date_Format>
</Csr>
```

Simply omit the element from a Create body to create the CSR without any roles.

Defaults

The following table lists the default values used, if certain elements are omitted from the create operation.

Timezone	GMT-5 (EST)
Status	1 (active)
Date_Format	mm/dd/yyyy
Role	None

Note: A CSR who has no role assigned will be unable to log into the Service Desk.

Date_Format Values

The following strings are the valid options for Date_Format:

mm/dd/yyyy
mm/dd/yy
dd/mm/yyyy
dd/mm/yy
month dd, yyyy

CSR Status

Similar to the CSR roles, the status of a CSR is an entity reference. You can List and Retrieve statuses. To set the value on a CSR use the ID of the specific time zone or status you want to use. To obtain a list, use
a request similar to the following:

```
https://my-sandbox.parature.com/api/v1/4139/4539/Csr/status?_token_
=QBgNnjmf1Tgt0GiqVENC3joCE...
```

You'll get a list of statuses as follows:

```
<entities total="5" results="5" page="1" page-size="25" href="https://my-
sandbox.parature.com/api/v1/4139/4539/Csr/status">
<Status id="1" uid="4139/4539/CsrStatus/1" href="https://sco-preview.
parature.com/api/v1/4139/4549/CsrStatus/1">

<Description display-name="Description" required="false" editable="true" data-
type="string">Regular user.</Description>
<Name display-name="Name" required="false" editable="true" data-
type="int">Active</Name>
</Status>
<Status id="-1" uid="4139/4539/CsrStatus/-1" href="https://scopreview.
parature.com/api/v1/4139/4549/CsrStatus/-1">
<Description display-name="Description" required="false" editable="
true" data-type="string"> Deactivated &#8211; no access
</Description>
<Name display-name="Name" required="false" editable="true" datatype="
int">Deactivated</Name>
</Status>
</entities>
```

CSR Time zone

Time zone is referenced as an entity type in the Parature CSR API. Use the listing call to get a list of available time zones.

```
https://my-sandbox.parature.com/api/v1/67/655/Timezone?_token_=WeWA3W2ESSH...
```

The list will look similar to:

```
<Entities total="34" results="25" page="1" page-size="25" href="https://my-
sandbox.parature.com/api/v1/59/200/Timezone">
...
<Timezone id="1" uid="59/200/Timezone/1" href="https://my-sandbox.
parature.com/api/v1/59/200/Timezone/1">
<Abbreviation display-name="Abbreviation" required="false" editable="true" data-
type="string">EST</Abbreviation>
<Timezone display-name="Time zone" required="false" editable="
true" data-type="string">Eastern Standard Time </Timezone>
</Timezone>
<Timezone id="5" uid="59/200/Timezone/5" href="https://my-sandbox.
parature.com/api/v1/59/200/Timezone/5">
<Abbreviation display-name="Abbreviation" required="false"
```

Password
As with the Customer entity, a CSR password is required in the Create operation. The password should be provided in the Update action only if you wish to change the existing value. The password will not be returned in List or Retrieve calls.

List CSRs
Use the status query parameter to list by CSR status id (user type). Currently, the only valid CSR status IDs are -1 (deactivated), and 1 (activated). Example:

```
https://my-sandbox.parature.com/api/v1/4/59/Csr?status_id_=1&_token_=WeWA3W2E...
```

Likewise, the time zone query parameter will search by time zone ID:

```
https://my-sandbox.parature.com/api/v1/4/59/Csr?timezone_id_=5&_token_=WeWA3W2E...
```

And finally, roles use the "role_id_" query parameter:

```
https://my-sandbox.parature.com/api/v1/4/59/Csr?role_id_=107,112&_token_=WeWA3W2E...
```

All other static fields can be queried with the standard API syntax.

Parature API Chat Object

First, read API Operations to understand how the API operations work for all objects. This page describes only those portions of the Chat API that are specific to the Chat object and assumes you are familiar with the default behavior.

NOTE: **ONLY** chats that have **ended** are visible through the API.

Supported Operations

The chat API is read-only and supports the Schema, Details, and List operations. In the schema all fields are set to editable=false since the API does not support Create or Update operations. Please read those sections for information on how to perform the operations.

Example 1 - List

List all chats in client 4139, department 4549

```
http://my-sandbox.parature.com/api/v1/4139/4549/Chat?_token_
=ABC123XYZ...
```

XML Output:

```
<Entities total="248" results="25" page="1" page-size="25" href="https://my-
sandbox.parature.com/api/v1/4139/4549/Chat?">
<Chat id="174610" uid="4139/4549/Chat/174610" href="https://my-
sandbox.parature.com/api/v1/4139/4549/Chat/174610" service-desk-uri="https://my-
sandbox.parature.com/ics/csrchat/History/ChatDetails.aspx?chatNum=174610">
<Browser_Language display-name="Browser Language" required="true" editable="false"
data-type="string">en</Browser_Language>
<Browser_Type display-name="Browser Type" required="true" editable="false" data-
type="string">Microsoft Internet Explorer</Browser_Type>
<Browser_Version display-name="Browser Version" required="true" editable="false"
data-type="string">8.0</Browser_Version>
<Chat_Number display-name="Chat #" required="false" editable="false" data-
type="string">174610</Chat_Number>
<Customer display-name="Customer" required="false" editable="false" data-
type="entity">
<Customer id="5006766" uid="4139/4549/Customer/5006766" href="https://my-
sandbox.parature.com/api/v1/4139/4549/Customer/5006766">
<First_Name display-name="First Name">Bob</First_Name>
</Customer>
</Customer>
<Date_Created display-name="Date Created" required="false" editable="false" data-
type="date">2009-03-30T18:46:11.000Z</Date_Created>
<Date_Ended display-name="Date Ended" required="false" editable="false" data-
type="date">2009-03-30T18:50:10.000Z</Date_Ended>
<Initial_Csr display-name="Initial CSR" required="false" editable="false" data-
type="entity">
<Csr id="22471" uid="4139/4549/Csr/22471" href="https://my-
sandbox.parature.com/api/v1/4139/4549/Csr/22471">
<Full_Name display-name="Full Name">Sam McClellan</Full_Name>
</Csr>
</Initial_Csr><Ip_Address display-name="IP Address" required="true" editable="false" data-
type="string">"10.20.8.39"</Ip_Address>
```

<Is_Anonymous display-name="Is Anonymous" required="true" editable="false" data-type="boolean">false</Is_Anonymous>
<Referrer_Url display-name="Referrer URL" required="true" editable="false" data-type="string"/>
<Related_Tickets display-name="Related Ticket Entities" required="false" editable="false" data-type="entity" multi-value="true">
<Ticket id="7342268" uid="4139/4549/Ticket/7342268" href="https://my-sandbox.parature.com/api/v1/4139/4549/Ticket/7342268">
<Ticket_Number display-name="Ticket #">4139-7342268</Ticket_Number>
</Ticket>
</Related_Tickets>
<Sla_Violations display-name="SLA Violations" required="false" editable="false" data-type="int">0</Sla_Violations>
<Status display-name="Status" required="false" editable="false" data-type="entity">
<Status id="1826" uid="4139/4549/Chat/status/1826" href="https://my-sandbox.parature.com/api/v1/4139/4549/Chat/status/1826">
<Name display-name="Name">Answered</Name>
</Status>
</Status>
<Summary display-name="Summary" required="false" editable="false" data-type="string">drerzt erzyr erz erzh 7342268 bye</Summary>
<User_Agent display-name="User Agent" required="true" editable="false" data-type="string">
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30618)"
</User_Agent>
</Chat>
...
</Entities>
.../api/v1/4139/4549/Chat?Initial_Csr_id_=22469&_token_=ABC123XYZ...

.../api/v1/4139/4549/Chat/1234?_token_=ABC123XYZ...
.../api/v1/4139/4549/Chat/1234/Transcript?_token_=ABC123XYZ...

Example 2 - List with specific criteria
List all chats with known initial Csr (id=22469)

Example 3A - Retrieve
Retrieve a chat object (id=1234) in

Example 3B – Retrieve Chat Transcript
Retrieve a chat transcript with chat object (id=1234)

Example 3C - Retrieve with History

Retrieve a chat object (id=1234) along with the chat object's history

```
.../api/v1/4139/4549/Chat/1234?_history_=true&_token_=ABC123XYZ...
```

XML Output:

```
<Chat id="174610" uid="4139/4549/Chat/174610" href="https://sco-
qa.parature.net/api/v1/4139/4549/Chat/174610" service-desk-uri="https://sco-
qa.parature.net/ics/csrchat/History/ChatDetails.aspx?chatNum=174610">

<Browser_Language display-name="Browser Language" required="true" editable="false"
data-type="string">en</Browser_Language>
<Browser_Type display-name="Browser Type" required="true" editable="false" data-
type="string">Microsoft Internet Explorer</Browser_Type>
<Browser_Version display-name="Browser Version" required="true" editable="false"
data-type="string">8.0</Browser_Version>
<Chat_Number display-name="Chat #" required="false" editable="false" data-
type="string">174610</Chat_Number>
<Customer display-name="Customer" required="false" editable="false" data-
type="entity">
<Customer id="5006766" uid="4139/4549/Customer/5006766" href="https://sco-
qa.parature.net/api/v1/4139/4549/Customer/5006766">
<First_Name display-name="First Name">Bob</First_Name>
</Customer>
</Customer>
<Date_Created display-name="Date Created" required="false" editable="false" data-
type="date">2009-03-30T18:46:11.000Z</Date_Created>
<Date_Ended display-name="Date Ended" required="false" editable="false" data-
type="date">2009-03-30T18:50:10.000Z</Date_Ended>
<Initial_Csr display-name="Initial CSR" required="false" editable="false" data-
type="entity">
<Csr id="22471" uid="4139/4549/Csr/22471" href="https://sco-
qa.parature.net/api/v1/4139/4549/Csr/22471">
<Full_Name display-name="Full Name">Sam McClellan</Full_Name>
</Csr>
</Initial_Csr>
<Ip_Address display-name="IP Address" required="true" editable="false" data-
type="string">"10.20.8.39"</Ip_Address>
<Is_Anonymous display-name="Is Anonymous" required="true" editable="false" data-
type="boolean">false</Is_Anonymous>
<Referrer_Url display-name="Referrer URL" required="true" editable="false" data-
type="string"/>
<Related_Tickets display-name="Related Ticket Entities" required="false"
editable="false" data-type="entity" multi-value="true">
<Ticket id="7342268" uid="4139/4549/Ticket/7342268" href="https://sco-
qa.parature.net/api/v1/4139/4549/Ticket/7342268">
<Ticket_Number display-name="Ticket #">4139-7342268</Ticket_Number>
</Ticket>
</Related_Tickets>
<Sla_Violations display-name="SLA Violations" required="false" editable="false"
data-type="int">0</Sla_Violations>
<Status display-name="Status" required="false" editable="false" data-
type="entity">
<Status id="1826" uid="4139/4549/Chat/status/1826" href="https://sco-
qa.parature.net/api/v1/4139/4549/Chat/status/1826">
<Name display-name="Name">Answered</Name>
```

```
</Status>
</Status>
<Summary display-name="Summary" required="false" editable="false" data-
type="string">drerzt erzyr erz erzh 7342268 bye</Summary>
<User_Agent display-name="User Agent" required="true" editable="false" data-
type="string">
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30618)"
</User_Agent>
<Custom_Field id="110909" display-name="Show other?" required="false" editable="true"
data-type="option" multi-value="false">

<Option id="232353">
<Value>Yes</Value>
<Enables>/Chat/Custom_Field[@id = 110885]</Enables>
<Enables>/Chat/Custom_Field[@id = 110889]</Enables>
<Enables>/Chat/Custom_Field[@id = 110890]</Enables>
<Enables>/Chat/Custom_Field[@id = 110891]</Enables>
<Enables>/Chat/Custom_Field[@id = 110892]</Enables>
<Enables>/Chat/Custom_Field[@id = 110893]</Enables>
<Enables>/Chat/Custom_Field[@id = 110898]</Enables>
<Enables>/Chat/Custom_Field[@id = 110899]</Enables>
<Enables>/Chat/Custom_Field[@id = 110900]</Enables>
<Enables>/Chat/Custom_Field[@id = 110901]</Enables>
<Enables>/Chat/Custom_Field[@id = 110902]</Enables>
<Enables>/Chat/Custom_Field[@id = 110910]</Enables>
</Option>
<Option id="232354" selected="true">
<Value>No</Value>
</Option>
</Custom_Field>
<ActionHistory>
<History id="833">
<Action id="2150" uid="4139/4549/Chat/action/2150" name="Route Chat"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2150"/>
<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="Rule"/>
<Action_Target display-name="Action Target" required="false" editable="false" data-
type="entity" target-type="Queue">
<Queue id="5228" uid="4139/4549/Queue/5228" href="https://sco-
qa.parature.net/api/v1/4139/4549/Queue/5228">
<Name display-name="Name">New Chat Queue</Name>
</Queue>
</Action_Target>
<To_Deflection display-name="To Deflection" required="false" editable="false" data-
type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false" data-
type="date">2009-03-30T18:46:20.000Z</Action_Date>
</History>
<History id="834">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>
<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="System"/>
<Action_Target display-name="Action Target" required="false" editable="false" data-
type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Yellow</Comments>
```

```
<To_Deflection display-name="To Deflection" required="false" editable="false"
data-type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false"
data-type="date">2009-03-30T18:46:52.000Z</Action_Date>
</History>
<History id="835">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>
<Action_Performer display-name="Action Performer" required="false"
editable="false" data-type="entity" performer-type="System"/>

<Action_Target display-name="Action Target" required="false" editable="false" data-
type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Red</Comments>
<To_Deflection display-name="To Deflection" required="false" editable="false" data-
type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false" data-
type="date">2009-03-30T18:47:22.000Z</Action_Date>
</History>
<History id="836">
<Action id="2158" uid="4139/4549/Chat/action/2158" name="Grab Chat"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2158"/>
<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="Csr">
<Csr id="22471" uid="4139/4549/Csr/22471" href="https://sco-
qa.parature.net/api/v1/4139/4549/Csr/22471">
<Full_Name display-name="Full Name">Sam McClellan</Full_Name>
</Csr>
</Action_Performer>
<Action_Target display-name="Action Target" required="false" editable="false" data-
type="entity" target-type="System"/>
<To_Deflection display-name="To Deflection" required="false" editable="false" data-
type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false" data-
type="date">2009-03-30T18:47:47.000Z</Action_Date>
</History>
<History id="837">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>
<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="System"/>
<Action_Target display-name="Action Target" required="false" editable="false"
data-type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Green</Comments>
<To_Deflection display-name="To Deflection" required="false" editable="false"
data-type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false"
data-type="date">2009-03-30T18:47:55.000Z</Action_Date>
</History>
<History id="838">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>
<Action_Performer display-name="Action Performer" required="false"
editable="false" data-type="entity" performer-type="System"/>
<Action_Target display-name="Action Target" required="false" editable="false"
data-type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Yellow</Comments>
```

```
<To_Deflection display-name="To Deflection" required="false" editable="false" data-
type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false" data-
type="date">2009-03-30T18:48:28.000Z</Action_Date>
</History>
<History id="839">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>

<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="System"/>
<Action_Target display-name="Action Target" required="false" editable="false" data-
type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Red</Comments>
<To_Deflection display-name="To Deflection" required="false" editable="false" data-
type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false" data-
type="date">2009-03-30T18:48:58.000Z</Action_Date>
</History>
<History id="840">
<Action id="2153" uid="4139/4549/Chat/action/2153" name="Change SLA"
href="https://sco-qa.parature.net/api/v1/4139/4549/Chat/action/2153"/>
<Action_Performer display-name="Action Performer" required="false" editable="false"
data-type="entity" performer-type="System"/>
<Action_Target display-name="Action Target" required="false" editable="false"
data-type="entity" target-type="System"/>
<Comments display-name="Comments" required="false" editable="false" data-
type="string">Green</Comments>
<To_Deflection display-name="To Deflection" required="false" editable="false"
data-type="boolean">false</To_Deflection>
<Action_Date display-name="Action Date" required="false" editable="false"
data-type="date">2009-03-30T18:49:51.000Z</Action_Date>
</History>
```

Action History

Since the Chat object tracks the lifecycle of the chat via Actions, the Chat Retrieve operation can optionally request the Action history along with the Chat details. By default, Action history is not returned in the Chat Retrieve. The _history_=true URI query parameter must be present in order for the history to be returned. Example below illustrates the proper way of retrieving Chat details with Action history.

Special Fields

Action Performer

This field represents the persona performing the action. The performer-type attribute specifies the type of persona performing the action. Below is a list of all possible performer-types:

- Anonymous
- CSR
- Customer
- Deployment
- Rule
- System

NOTE: Only CSR and Customer performer-types are displayed as auxiliary (second-order) objects. Example with a CSR performer-type:

```
<History id="123">
...
<Action_Performer display-name="Action Performer"
required="false" editable="false" data-type="entity" performertype="

Csr">
<Csr id="22471" uid="4139/4549/Csr/22471" href="https://scoqa.
parature.net/api/v1/4139/4549/Csr/22471">
<Full_Name display-name="Full Name">Sam McClellan</Full_Name>
</Csr>
</Action_Performer>
...
</History>
Example with a system performer-type:
<History id="123">
...
<Action_Performer display-name="Action Performer"
required="false" editable="false" data-type="entity" performertype=" System"/>
...
</History>
```

Action Target

This field represents the persona 'receiving' the action, for example the Queue to which the chat was routed. The target-type attribute specifies the type of persona targeted the action.

- CSR
- Queue
- System

NOTE: Only Csr and Queue target-types are displayed as auxiliary (second-order) objects. Example with a queue target-type:

```
<History id="123">
...
<Action_Target display-name="Action Target" required="false" editable=" false" data-
type="entity" target-type="Queue">
<Queue id="5228" uid="4139/4549/Queue/5228" href="https://scoqa.
parature.net/api/v1/4139/4549/Queue/5228">
<Name display-name="Name">New Chat Queue</Name>
</Queue>
</Action_Target>
...
</History>
```

Example with a system target-type:

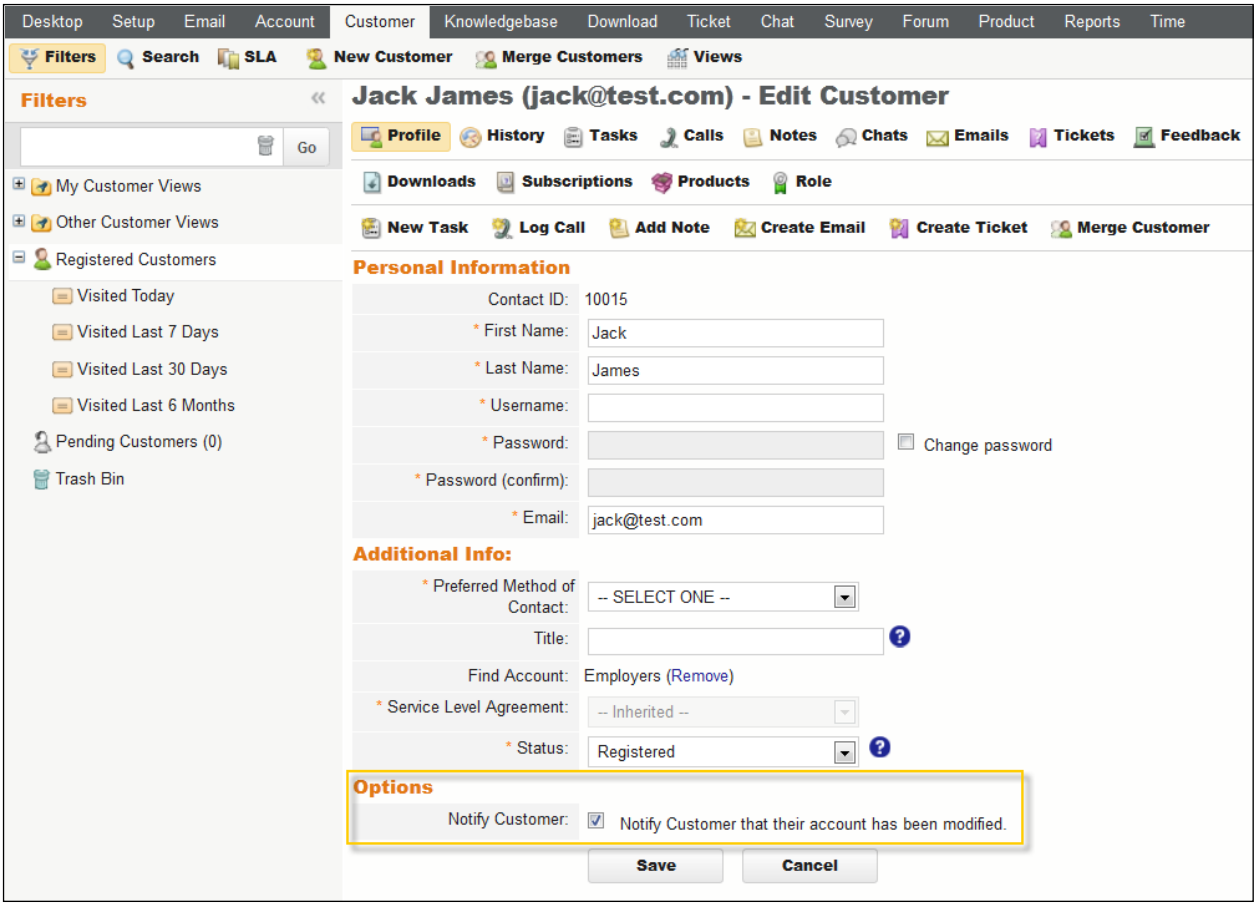
```
<History id="123">
...
<Action_Target display-name="Action Target" required="false" editable="
false" data-type="entity" target-type="System"/>
...
</History>
```

Parature API Customer Object

First, read API Operations to understand how the API operations work for all objects. This page describes only those portions of the Customer API that are specific to the Customer object and assumes you are familiar with the default behavior.

Notifying Customers of Password Change

When creating or updating Customer objects in the Service Desk, users can indicate whether or not to notify the customer that their account has been modified.



Customer notification can also be performed via the API using specialized URI query-string parameters when performing the API Create or Update operation on a Customer object. The parameters are _notify_ and _includePassword_. The default value for each of these parameters is false, which means that if either parameter is **not** included in the URI request, the notification will not occur. The following is a sample Customer request URI for a create or update operation in which the customer notification is enabled:

```
https://my-servicedesk.parature.com/api/v1/234/567/Customer?_notify_=true&_includePassword_=false&_token_=I7GNr0G7ngMz...
```

Customer Role field

Role entity

The "second-order" object type called (Customer) **Role** supports only the **Schema**, **List**, and **Retrieve** operations and its uri base location is .../Customer/role. The following example shows the List operation for customer roles:

```
https://myfarm-sandbox.parature.com/api/v1/33/420/Customer/role?_token_=ABC123XYZ...
```

Note: Be sure to rely only on the id's of the roles for relational purposes; their names are subject to change.

Customer Role field

The **Customer Role** field is an entity field that determines what role is assigned to the customer.

```
<Customer_Role display-name="Default Customer Role" required="false" editable="true" data-type="entity">
```

Example:

```
<Customer>
...
<Customer_Role display-name="Default Customer Role" required="false" editable="true" data-type="entity">
<CustomerRole id="3252" uid="4/52/Customer/role/3252" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Customer/role/3252">
<Name display-name="Name">Consultant</Name>
</CustomerRole>
</Customer_Role>
</Customer>
```

Values
Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation. The following Customer Roles are available:

- Account Administrator
- Account Reviewer
- Consultant
- Customer
- Customer (Limited)

See the Parature documentation for full details of these roles and their meanings.

Note: The Consultant role is only available in the Parature API if the Parature Consultant feature has been enabled for your Parature software. Please contact your Account Executive if you have any questions about this feature.

Defaults
If no Customer_Role value is specified while creating a new Customer, a default will be assigned. If the Customer is associated with an Account, the Default_Customer_Role will be used (see Account Object Changes). If the Customer is NOT associated with an Account, their role will default to the "Customer" role.

Ticket Object changes

Note: API changes pertaining to the Additional Contact field will only be available in the Parature API if the Parature Consultant feature has been enabled for your Parature software. Please contact your Account Executive if you have any questions about this feature.

Additional Contact field
The **Additional Contact** is an entity field that relates to the secondary customer on the ticket. The field is structurally similar to **Ticket_Customer** field and its href attribute can be used to retrieve more information about the additional customer.

Example:

```
<Ticket>
... other ticket fields ...
<Additional_Contact display-name="Additional Contact" required="false"
editable="false" data-type="entity">
<Customer id="1234" uid="4/52/Customer/1234" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Customer/1234">
<First_Name display-name="First Name">Fred</First_Name>
</Customer>
</Additional_Contact>
...
</Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Additional Contact Account filter

A filter was added to the ticket **List** operation for the additional contact's Account id, and it behaves very similar to the filter by primary customer's Account id (ticket_account_id_). The keyword used to filter tickets by the Additional_Contact's account is additional_account_id. The following example shows the List operation for tickets using this new filter. The example will filter the Ticket list to only those tickets where the Additional Contact belongs to the Account with id 5.

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?additional_contact_account_id_=5&_token_=AB
C123XYZ...
```

Note: Sorting by this field on ticket lists is not available.

Show Action To Additional Contact

This is a change to the Action sub-element of the Ticket object. When performing an action on a Ticket object via the API, the ShowToAdditionalContact boolean controls whether the action being performed will be visible to the Additional_Contact on the ticket. If this value is false, this action will not be visible to the Additional_Contact when they view the Ticket on the portal.

```
<ShowToAdditionalContact display-name="ShowToAdditionalContact" required="false"
editable="true" data-type="boolean"/>
```

See below for an example that sets it to false for the action.

```
<Ticket>
<Action>
<Action id="482">
<AssignToCsr>26</AssignToCsr>
<Comment>Assigning the ticket to you. Have fun!</Comment>
<ShowToCust>false</ShowToCust>
<ShowToAdditionalContact>false</ShowToAdditionalContact>
<TimeSpentMinutes>15</TimeSpentMinutes>
</Action>
</Action>
</Ticket>
```

Default Show Value

If the ShowToAdditionalContact field is not included in the action API the action will not be visible to the Additional Contact on the portal.

Email Notification Additional Contact field

The **Email Notification Additional Contact** field is a Boolean flag that tells whether the Additional Contact on the ticket will receive email notifications about updates to the ticket. Note that when set to true, email notifications to the Additional Contact will only be sent if the action also specifies ShowToAdditionalContact=true (both if the action occurs from the Service Desk or via the API). This field is in the following format in the API schema:

```
<Email_Notification_Additional_Contact display-name="Email Notification Additional
Contact" required="false" editable="true" data-type="boolean">
```

The example below shows the value being set to true.

```
<Ticket>
... other ticket fields
<Email_Notification_Additional_Contact>true</Email_Notification_Additional_Contact></
Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Hide From Customer field

The **Hide From Customer** field is a boolean flag that tells whether the ticket is hidden from primary customer.

Example:

```
<Ticket>
...
<Hide_From_Customer display-name="Hide from customer" required="false"
editable="true" data-type="boolean">false</Hide_From_Customer>
...
</Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Note: If Hide_From_Customer field is set to true, the ticket will not be visible to the primary customer. However, it will still be visible to the Secondary_Contact.

Parature Asset & Product API

First read about API Operations to understand how the API operations work for all objects. This page describes only those portions of the Product and Asset API that are specific to the Product or Asset objects and assumes you are familiar with the default behavior.

To use these APIs it's important to understand the relationship between the Product and Asset objects. A Product is typically a type or class of something, and an Asset is a single instance or ownership of that something. For example, a Product might be a particular type or model of digital camera. An Asset would be an actual physical camera, of that model, owned by a person (Customer) or company (Account). Product ownership (via an Asset) can be used on the portal to filter a user's view of information.

Your organization's Parature support software can be configured in three different modes of Product and Asset behavior. The most common mode is *Full Product and Asset* mode, where the service desk gives direct access to both products and assets. In *Simple Product* mode, the service desk shows only Products. And in *Asset Only* mode the service desk displays only Assets. You will use the Product and Asset APIs differently depending on which configuration your organization uses. If you are not sure which configuration is in use contact your Parature administrative user or contact Parature customer support at the Parature Support Portal (https://support.parature.com)

The Asset API behaves differently based on configuration. Follow the guides below for the Product/Asset configuration in use.

Full Product and Asset Mode

In the Full Product and Asset mode your service desk users can see both the product catalog and the assets. You can use the Product API to view, create, update and delete Products within your system. You can use the Asset API to create asset "instances" of your Products. You can then view, update or delete those Assets. When creating an asset you must supply both the Product the Asset is associated with, and the Customer or Account that owns the Asset.

- In this mode, both Products and Asset can contain custom fields defined for your organization. These fields often carry information specific to your use of Products and Assets. A Schema operation on these objects will reveal the custom fields defined.

- In this mode, you cannot assign the Asset a name, as its name is always the same as the name of the Product of which it is an instance.

- Asset workflow is supported in this mode, and you can perform actions on Assets via the API. Although for assets, the workflow only changes the Asset status, which can also be performed using an Update API call.

To assign an product to an owner

1. If the Product doesn't already exist, create it.
2. Create an Asset with a Product field set to the ID of the Product, and the Owner field set to the Customer ID or Account ID to which you are assigning the Product.

To reassign ownership of a Product to a different Customer or Account

1. Update the Asset, setting the Owner field to the new Customer or Account.

Simple Product Mode

In Simple Product mode, your service desk users can see only Products. Although the users cannot see Assets from the service desk, they do actually exist. The Assets serve to create the relationship between a Product and the Customer or Account that owns it. So, although the service desk users cannot see or edit the Assets, you must create an Asset via the API in order to assign a Product to a Customer/Account.

- In this mode Assets can only be created or deleted; they cannot be updated.
- A key difference in Simple Product mode is that the Asset does not contain any additional custom fields. Asset contains only the fields necessary to maintain the Product / owner relationship.

To assign ownership of a Product to a Customer or Account

1. If the Product doesn't already exist, create it.
2. Create an Asset with a Product field set to the ID of the Product, and the Owner field set to the Customer ID or Account ID to which you are assigning the Product.

To reassign ownership of a Product to a different Customer or Account
This cannot be done in Simple Product mode. Instead,

1. Delete the old Asset
2. Create a new Asset linking the Product to the new owner, as defined above.

Asset-Only Mode

In Asset-Only mode no Products exist. Each Asset is created separately and is not an instance of a Product, but represents a separate/unique object. When assets are created an Owner, either Customer or Account, must be supplied.

- Product does not exist in this mode. Because of this, each Asset is unique and distinct from every other asset, and do not fall into controlled categories.

- Asset workflow is supported in this mode, and you can perform actions on Assets via the API. Although for assets, the workflow only changes the Asset status, which can also be performed using an Update API call.

To assign an Asset to a Customer or Account:

1. Create the Asset, leaving the Product field blank, and setting the Owner field to the ID of the Customer or Account to which you are assigning the Asset.

To reassign an Asset to a new owner

1. Update the Asset, setting the Owner field to the new Customer or Account ID.

Products, Assets and Trashing

On the service desk Assets appear to have the ability to trash separately from purge. However, they do not have true trash behavior. Instead, a simplified form of workflow may be enabled. This is not equivalent to trashing and the *purge* option is used with Assets.

Product Folders

A ProductFolder is a secondary business object within the Parature system. Folders can be managed vi a the API, allowing all the normal list, retrieve, create, update and delete operations. A simple example is shown below.

List Product Folders

The ProductFolder object type is used to retrieve a list of all folders within the Product module.

```
https://my-sandbox.parature.com/api/v1/4100/4549/ProductFolder?_token_=SOZhBs...
<Entities total="1" results="1" page="1" page-size="25"
href="http://localhost/api/v1/4/52/ProductFolder">
<ProductFolder id="194330" uid="4100/4549/ProductFolder/19430" href="https://my-
sandbox.parature.com/api/v1/4100/4549/ProductFolder/19430">
<Is_Private display-name="Is Private" required="false" editable="true" data-
type="boolean">false</Is_Private>
<Description display-name="Description" required="false" editable="true" data-
type="string"/>
<Name display-name="Name" required="true" editable="true" data-
type="string">Top Level Folder</Name>
<Date_Updated display-name="Date Updated" required="false" editable="false"
data-type="date">2006-07-10T17:14:09.220Z</Date_Updated>
<Parent_Folder display-name="Parent Folder" required="true" editable="true"
data-type="entity">
<ProductFolder id="1022" uid="4100/4549/ProductFolder/1022"
href="https://my- sandbox.parature.com/api/v1/4100/4549/ProductFolder/1022">
<Name display-name="Name">My Products</Name>
</ProductFolder>
</Parent_Folder>
</ProductFolder>
...
<Entities>
```

Retrieving a Product Folder

To retrieve the details of a folder object, the Product Folder object type and ID are used in the URI of the retrieve/GET command as follows:

```
https://my-
sandbox.parature.com/api/v1/4100/4549/ProductFolder/8675309?_token_=SOZhBs...
```

Creating a Product Folder

A ProductFolder can be created using a create/POST command.

https://my-sandbox.parature.com/api/v1/4100/4549/ProductFolder?_token_=SOZhBs...

The request body must contain the required information for creating the Product Folder.

```
<ProductFolder>
<Is_Private>false</Is_Private>
<Description>My new folder for testing</Description>
<Name>ETest</Name>
<Parent_Folder>
<ProductFolder id="19430" \>
</Parent_Folder>
</ProductFolder>
```

Updating a Product Folder

A ProductFolder can be updated using an update/PUT command.

https://my- sandbox.parature.com/api/v1/4100/4549/ProductFolder/12345?_token_=SOZhBs...

The request body must contain all information for the ProductFolder. Any optional fields that are not submitted in the body are set to null/blank values.

```
<ProductFolder>
<Is_Private>true</Is_Private>
<Description>My new folder for testing - updated the description</Description>
<Name>ETest</Name>
<Parent_Folder>
<ProductFolder id="19430" \>
</Parent_Folder>
</ProductFolder>
```

Deleting a Product Folder

Unlike most of the first order obejcts, such as Customers, Tickets, and Products, folders do not have a "trashed" state from which they can be purged or untrashed. Folders can only be purged. However, before the API purges a folder, a check is performed to determine if the folder is empty. Only empty folders can be purged. Attempting to purge a folder that contains any objects or subfolders will result in an error. To purge a ProductFolder, the Delete operation is performed using the following URI:

```
https://my-
sandbox.parature.com/api/v1/4100/4549/ProductFolder?_purge_=true&_token_=SOZhBs...
```

Parature API Ticket Object

First, read API Operations to understand how the API operations work for all objects. This page describes only those portions of the API that are specific to the Ticket object and assumes you are familiar with the default behavior.

Attachments

Adding Attachments

File attachments can be associated with a Ticket object through the API. The file attachment process is a multi-step process.

- Step 1 : Get an upload URI
  - A file upload URI must be obtained from the API and used to upload the attachment file.
  - The upload URI contains a temporary token that expires after 90 minutes; therefore, the upload URI should only be used for a single attachment operation.
- Step 2 : Upload a file
  - When performing the file upload, an HTTP Post method is used to post the files.
- Step 3 : Attach files to the Ticket
  - After the attachment files are successfully uploaded, the API responds with an XML result object containing unique identifiers for each of the uploaded files.
  - These unique identifiers, or GUIDs, are used within a Ticket create/update operation to associate the attachment files with the Ticket object.

The following is an example of the process to attach a file to a Ticket object:

Step 1: Get an upload URI URI Request:

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/upload?_token_=VKWHzhBGSe2...
```

XML Output:

```
<Upload href="https://my-
sandbox.parature.com/FileManagement/Upload/?token=Bp3sRV@8WCFBG...&amp;UploadID=501
550843 "/>
```

Note that the href value includes an XML escaped ampersand character (&). This character must be un-escaped when using the upload URI so that the URI is as follows:

```
.../FileManagement/Upload/?token=Bp3sRV@8WCFBG...&UploadID=501550843
```

Step 2: Upload the file using the Upload URI

For the sake of simplicity, the cURL utility is used to demonstrate a file upload in this example. cURL is a command line utility for transferring files with URL/URI syntax.

```
>curl -F "file=@MyAttachment.jpg" "https://myfarm-
sandbox.parature.com/FileManagement/Upload/?token=Bp3sRV@8WCFBG...&UploadID=5
01550843"
```

You may also use a simple HTML form, with a file input, to upload the file.

In either way, you should get an XML similar to the one below that provides you with a GUID. XML Output:

```
<result>
<passed>
<file>
<inputname>file</inputname>
<filename>MyAttachment.jpg</filename>
<guid>172abe4de93843debfa323976d8e1b91</guid>
</file>
</passed>
</result>
```

Step 3: Update the Ticket with the file attachment GUID URI Request: https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/67?_token_=VKWHzhBGSe2...
XML Input:

```
<Ticket>
... Other ticket fields ...
<Ticket_Attachments>
<Attachment>
<Guid>172abe4de93843debfa323976d8e1b91</Guid>
<Name>MyAttachment.jpg</Name>
</Attachment>
</Ticket_Attachments>
</Ticket>
```

Just like the Service Desk, you can attach up to 10 attachments to a ticket. In the case that you want to attach more than 1 file to a Ticket, simply add another Attachment tag to the XML.

For example,

```
<Ticket>
... Other ticket fields ...
<Ticket_Attachments>
<Attachment>
<Guid>172abe4de93843debfa323976d8e1b91</Guid>
<Name>MyAttachment1.jpg</Name>
</Attachment>
<Attachment>
<Guid>172ab2jf8j983joweofw23976d8e1b91</Guid>
<Name>MyAttachment2.jpg</Name>
</Attachment>
</Ticket_Attachments>
</Ticket>
```

In a Ticket Retrieve operation, you will get a list of attachments back. To add an attachment, repeat step 1 & 2 to get the attachment GUID, and add an Attachment tag to the XML.

For example, your Retrieve operation gives you back XML similar to the following:

```
<Ticket id="123">
... Other ticket fields ...
<Ticket_Attachments>
<Attachment>
<Guid>172abe4de93843debfa323976d8e1b91</Guid>
<Name>MyAttachment1.jpg</Name>
</Attachment>
<Attachment>
<Guid>172ab2jf8j983joweofw23976d8e1b91</Guid>
<Name>MyAttachment2.jpg</Name>
</Attachment>
</Ticket_Attachments>
</Ticket>
```

After repeating step 1 & 2, 123456789 is your new file GUID. Then, in the Retrieve XM L, add the attachment tag to the XML, and update the ticket through the API Update operation.

```
<Ticket id="123">
... Other ticket fields ...
<Ticket_Attachments>
<Attachment>
<Guid>172abe4de93843debfa323976d8e1b91</Guid>
<Name>MyAttachment1.jpg</Name>
</Attachment>
<Attachment>
<Guid>172ab2jf8j983joweofw23976d8e1b91</Guid>
<Name>MyAttachment2.jpg</Name>
</Attachment>
<Attachment>
<Guid>123456789</Guid>
<Name>NewAttachment.txt</Name>
</Attachment>
</Ticket_Attachments>
</Ticket>
```

Removing Attachments

To remove an attachment, simply remove the corresponding Attachment tag from the XML, and do an Update operation.

Downloading Attachments
To download an attachment, there are 3 options:

1. Use the href attribute of the Attachment element. Do not distribute this url, as anyone can download the file using this url, but this is the preferred download option for programmatic access.
2. Use the secure-service-desk-url attribute of the Attachment element. This url should be used for browser access and requires a Service Desk login and access to the ticket.
3. Use the secure-portal-url attribute of the Attachment element. This url should be used for browser access and requires a customer portal login and access to the ticket.

Here is an example of the values:

```
<Ticket id="123">
... Other ticket fields ...
<Ticket_Attachments>
<Attachment href="https://my-
sandbox.parature.com/FileManagement/Download/4d1b7fb220c548a2a11a015bbc95eb1e?token
=abc" secure-service-desk-url="https://my-
sandbox.parature.com/link/desk/14/62/Ticket/123/attachment/4d1b7fb220c548a2a11a015b
bc95eb1e" secure-portal-url="http://my-
sandbox.parature.com/link/portal/14/62/Ticket/123/attachment/4d1b7fb220c548a2a11a01
5bbc95eb1e">
<Guid>172abe4de93843debfa323976d8e1b91</Guid>
<Name>MyAttachment1.jpg</Name>
</Attachment>
</Ticket_Attachments>
</Ticket>
```

Special Fields

The Ticket object has some fields with non-standard or notable behaviors.

Ticket_Account Filter

Ticket lists can be filtered by a value called Ticket_Account. Filtering by this value returns a list of tickets where the primary contact belongs to a specific Account. While Ticket_Account is not an actual data field on the ticket object, and will not appear in the schema, the filter for Ticket_Account follows the field- filtering format. To use Ticket_Account simply add it to the ticket list URL as a parameter, e.g.

```
https://my-servicedesk.parature.com/api/v1/123/456/Ticket?Ticket_Account_id_=18254&_token_=I7GNr0G7n gMz...
```

In this example, the list would return only those tickets associated with the account with id=18254

Additional Contact field

The **Additional Contact** is an entity field that relates to the secondary customer on the ticket. The field is structurally similar to **Ticket_Customer** field and its href attribute can be used to retrieve more information about the additional customer.
Example:

```
<Ticket>
... other ticket fields ...
<Additional_Contact display-name="Additional Contact" required="false" editable="false" data-type="entity">
<Customer id="1234" uid="4/52/Customer/1234" href="https://myfarm-sandbox.parature.com/api/v1/4/52/Customer/1234">
<First_Name display-name="First Name">Fred</First_Name>
</Customer>
</Additional_Contact>
...
</Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Additional Contact Account Filter

A filter is available on the ticket List operation for the additional contact's Account id. It behaves very similar to the filter by primary customer's Account id and it's called Additional_Contact_Account. While it is not an actual data field on the Ticket object, and will not appear in the schema, the filter for Additional_Contact_Account follows the field-filtering format. The following example shows the List operation for tickets using this new filter. The example will filter the Ticket list to only those tickets where the Additional Contact belongs to the Account with id 5.

```
https://myfarm-sandbox.parature.com/api/v1/33/420/Ticket?additional_contact_account_id_=5&_token_=ABC123XYZ...
```

Note: Sorting by this field on ticket lists is not available.

Ticket_Product

For certain configurations of Parature, Tickets can be associated with Products. Within the Service Desk, this field will display as a dropdown called Product. However, as discussed in the Product/Asset API, these are actually Asset objects which link a customer, account, or ticket to a product. The field displays as Ticket_Asset via the API, as shown in the example below.

```
<Ticket_Asset display-name="Product" required="false" editable="true" data-
type="entity">
<Asset id="169624" uid="4138/4546/Asset/169624" href="https://sco-
sandbox.parature.com/api/v1/4138/4546/Asset/169624">
<Name display-name="Name" />
</Asset>
</Ticket_Asset>
```

Visibility and Notification of Tickets and Ticket Changes

A number of options exist that allow control of how a contact, or secondary contact, is notified of changes to a ticket, and whether those changes or comments are visible to the contacts. The fields or flags which control this visibility include whether email notifications are sent to the customer, or the additional customer, and the show-to flags, which control whether the action/change is visible on the portal, for both primary and additional customer.

The **Email_Notification** field on the Ticket is being changed to the original default of **FALSE** if not provided.

Change Visibility Flags for Primary Customer

	Email Notification Field	
	On	Off
ShowToCustomer = True	Action is shown on portal, email is sent	Action is shown on portal, email is NOT sent
ShowToCustomer = False	Action is hidden on portal, email is NOT sent	Action is hidden on portal, email is NOT sent

	Email Notification Additional Contact	
	On	Off
ShowToAdditionalContact = True	Action is shown on portal, email is sent	Action is shown on portal, email is NOT sent
ShowToAdditionalContact = False	Action is hidden on portal, email is NOT sent	Action is hidden on portal, email is NOT sent

Email Notification Additional Contact field

The **Email Notification Additional Contact** field is a boolean flag that tells whether the Additional Contact on the ticket will receive email notifications about updates to the ticket. Note that when set to true, email notifications to the Additional Contact will only be sent if the action also specifies ShowToAdditionalContact=true (both if the action occurs from the Service Desk or via the API). This field is in the following format in the API schema:

```
<Email_Notification_Additional_Contact display-name="Email Notification Additional
Contact" required="false" editable="true" data-type="boolean">
```

The example below shows the value being set to true.

```
<Ticket>
... other ticket fields ...
<Email_Notification_Additional_Contact>true</Email_Notification_Additional_Contact>

</Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Show Action To Additional Contact

When performing an action on a Ticket object via the API, the ShowToAdditionalContact boolean controls whether the action being performed will be visible to the Additional_Contact on the ticket. If this value is false, this action will not be visible to the Additional_Contact when they view the Ticket on the portal.

```
<ShowToAdditionalContact display-name="ShowToAdditionalContact" required="false"
editable="true" data-type="boolean"/>
```

See below for an example that sets it to false for the action.

```
<Ticket>
<Action>
<Action id="482">
<AssignToCsr>26</AssignToCsr>
<Comment>Assigning the ticket to you. Have fun!</Comment>
<ShowToCust>false</ShowToCust>
<ShowToAdditionalContact>false</ShowToAdditionalContact>
<TimeSpentMinutes>15</TimeSpentMinutes>
</Action>
</Action>
</Ticket>
```

Default Show Value

If the ShowToAdditionalContact field is not included in the action API the action will not be visible to the Additional Contact on the portal.

Hide From Customer field

The **Hide From Customer** field is a boolean flag that tells whether the ticket is hidden from primary customer. If a ticket is hidden, the primary contact will not be able to view the ticket on the portal, it will not appear in the contqact's ticket lists, and the contact will not receive email notification for changes to that ticket.*Note: This flag does not affect visibility of the ticket to the secondary contact*

Example:

```
<Ticket>
...
<Hide_From_Customer display-name="Hide from customer" required="false"
editable="true" data-type="boolean">false</Hide_From_Customer>
...
</Ticket>
```

Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Parent/Child Ticket Relationships

Ticket objects can be associated with each via a parent/child relationship. Each Ticket can have at most one parent Ticket, but can have multiple child Tickets. Both the parent and child fields are entity type fields, which means they point to another Parature object; Ticket in this case. Notice in the sample below that Ticket_Children has the attribute multi-value=true.

```
<Ticket_Parent display-name="Parent Ticket" required="false" editable="true" data-
type="entity">
<Ticket id="5114304" uid="123/456/Ticket/5114304" href="https://my-
servicedesk.parature.com/api/v1/123/456/Ticket/5114304">
<Ticket_Number display-name="Ticket #">123-5114304</Ticket_Number>
</Ticket>
</Ticket_Parent>
<Ticket_Children display-name="List of child tickets" required="false"
editable="false" data-type="entity" multi-value="true">
<Ticket id="5151760" uid="123/456/Ticket/5151760" href="https://my-
servicedesk.parature.com/api/v1/123/456/Ticket/5151760">
<Ticket_Number display-name="Ticket #">123-5151760</Ticket_Number>
</Ticket>
<Ticket id="5158900" uid="123/456/Ticket/5158900" href="https://my-
servicedesk.parature.com/api/v1/123/456/Ticket/5158900">
<Ticket_Number display-name="Ticket #">123-5158900</Ticket_Number>
</Ticket>
</Ticket_Children>
```

Parent/Child Behavior on Updates

When updating a ticket where you are adding, removing or modifying a parent/child ticket relationship, the API behaves as follows:

API Update Request	Resulting Action

Ticket Copy/Clone

An operation or action for cloning a Ticket via the API does not exist. Ticket cloning can be performed by a two-step process of retrieving the details for a given Ticket and submitting an API Create operation with these fields to produce a copy of the original Ticket.

Actions

In addition to the standard API operations that are available to all objects, the Ticket object also allows Actions to be executed on a Ticket instance. Actions are additional object transitions that can occur in which object fields or state can be changed. For example, assigning a Ticket to a CSR or changing the status of a ticket from "work in progress" to "resolved". Actions are performed on a Ticket using the API Update operation.

Since Actions are dependent upon the current status of a Ticket, the process of performing an Action requires three steps. The list of permitted Actions for a given Ticket in its current state are obtained by retrieving the Ticket details. Using the list of Actions, an Action schema object is requested to obtain a list of the fields associated with a given action. From the Action schema, an Action XML object is constructed and inserted into a Ticket XML object, which is used when performing the Ticket Update operation. The following is an example of the process to perform the "assign ticket" Action to assign a Ticket to a CSR:

parature

Step 1: Get the Ticket details

URI Request:

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/66?_token_=VKWHzhBGSe2...
```

XML Output:

```
<Ticket id="66" uid="14/62/Ticket/66" href="https://myfarm-
sandbox.parature.com/api/v1/14/62/Ticket/66" service-desk- uri="https://myfarm-
sandbox.parature.com/ics/tt/ticketDetail.asp?ticket_id=66">
<TicketNumber display-name="TicketNumber" required="false" editable="false"
data-type="string">14-66</TicketNumber>
<Status display-name="Status" required="true" editable="false" data-
type="entity">
<Status id="71" uid="14/62/Status/71"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/Status/71">
<Name display-name="Name">Open</Name>
<CSRText display-name="CSRText">New</CSRText>
<CustomerText display-name="CustomerText">Open</CustomerText>
</Status>
</Status>
<Customer display-name="Customer" required="true" editable="true" data-
type="entity">
<Customer id="23" uid="14/62/Customer/23"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/Customer/23" service-desk-
uri="https://myfarm-sandbox.parature.com/api/v1/14/62/Customer/23">
<Full_Name display-name="Full Name">Customer Sample</Full_Name>
</Customer>
</Customer>
<Queue display-name="Queue" required="false" editable="false" data-type="entity">
<Queue id="11" uid="14/62/Queue/11" href="https://myfarm-
sandbox.parature.com/api/v1/14/62/Queue/11">
<Name display-name="Name">New Ticket Queue</Name>
</Queue>
</Queue>
<DateCreated display-name="Date Created" required="false" editable="false" data-
type="date">2007-08-12T17:19:03.177</DateCreated>
<DateUpdated display-name="Date Updated" required="false" editable="false" data-
type="date">2007-08-12T17:19:03.177</DateUpdated>
...
<Actions>
<Action id="482" uid="62/Ticket/action/482" name="Assign Ticket"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/action/482"/>
<Action id="490" uid="14/62/Ticket/action/490" name="Grab Ticket"
href="https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/action/490"/>
</Actions>
</Ticket>
```

Step 2: Get the Action schema for Assign Ticket action

URI Request:

```
https://myfarm-
sandbox.parature.com/api/v1/14/62/Ticket/action/482?_token_=VKWHzhBGSe2.
..
```

XML Output:

```
<Action>
<Name display-name="Name" required="false" editable="false" data-
type="string">Assign Ticket</Name>
<Type display-name="Type" required="false" editable="false" data-
type="string">Assign</Type>
<Comment display-name="Comment" required="false" editable="true" data-
type="string"/>
<ShowToCust display-name="ShowToCust" required="false" editable="true" data-
type="boolean">false</ShowToCust>
<EmailText display-name="EmailText" required="false" editable="true" data-
type="string">You have been assigned the following ticket</EmailText>
<EmailCsrList display-name="EmailCsrList" required="false" editable="true" data-
type="string"/>
<EmailCustList display-name="EmailCustList" required="false" editable="true"
data-type="string"/>
<TimeSpentHours display-name="TimeSpentHours" required="false" editable="true"
data-type="int">0</TimeSpentHours>
<TimeSpentMinutes display-name="TimeSpentMinutes" required="false"
editable="true" data-type="int">0</TimeSpentMinutes>
<AssignToCsr display-name="AssignToCsr" required="true" editable="true" data-
type=""/>
</Action>
```

Step 3: Perform the action

URI Request:

```
https://myfarm-sandbox.parature.com/api/v1/14/62/Ticket/66?_token_=VKWHzhBGSe2...
```

XML Input:

```
<Ticket>
<Action>
<Action id="482">
<AssignToCsr>26</AssignToCsr>
<Comment>Assigning the ticket to you. Have fun!</Comment>
<ShowToCust>false</ShowToCust>
<TimeSpentMinutes>15</TimeSpentMinutes>
</Action>
</Action>
</Ticket>
```

Show Action To Additional Contact

When performing an action on a Ticket object via the API, the ShowToAdditionalContact boolean controls whether the action being performed will be visible to the Additional_Contact on the ticket. If this value is false, this action will not be visible to the Additional_Contact when they view the Ticket on the portal.

```
<ShowToAdditionalContact display-name="ShowToAdditionalContact" required="false"
editable="true" data-type="boolean"/>
```

See below for an example that sets it to false for the action.

```
<Ticket>
<Action>
<Action id="482">
<AssignToCsr>26</AssignToCsr>
<Comment>Assigning the ticket to you. Have fun!</Comment>
<ShowToCust>false</ShowToCust>
<ShowToAdditionalContact>false</ShowToAdditionalContact>
<TimeSpentMinutes>15</TimeSpentMinutes>
</Action>
</Action>
</Ticket>
```

Default Show Values

If the ShowToCust field is not included in the action API the action will not be visible to the Customer on the portal.

If the ShowToAdditionalContact field is not included in the action API the action will not be visible to the Additional Contact on the portal.

Action History

Since the Ticket object supports execution of Actions, the Ticket Retrieve operation can optionally request the Action history along with the Ticket details. By default, Action history is not returned in the Ticket Retrieve. The _history_=true URI query parameter must be present in order for the history to be returned. The following is an example URI for retrieving Ticket details with Action history:

Ticket Status

As a Ticket is processed through its workflow of various actions, the current status of the Ticket is reflected in the Ticket_Status field. The status field displays the name of the status and an XML Object Location for the status details, as follows:

```
<Ticket_Status display-name="Status" required="false" editable="false" data-
type="entity">
<Status id="1" uid="4/52/Ticket/status/1" href="https://my-
servicedesk.parature.com/api/v1/4/52/Ticket/status/1">
<Name display-name="Name">New</Name>
</Status>
</Ticket_Status>
```

The status details show the Name of the status as it would be displayed to a customer service representative and the "customer text", which is the name of the status as it would be displayed to customers on the support portal. In addition, the Status contains a status-type attribute that indicates if the status represents a ticket that is in either an open or closed state. The following is an example of the Ticket Status details:

```
<Status id="1" uid="4/52/Ticket/status/1" href="https://my-
servicedesk.parature.com/api/v1/4/52/Ticket/status/1" status-type="open">
<Customer_Text display-name="Customer Text" required="true" editable="true" data-
type="string">Open</Customer_Text>
<Name display-name="Name" required="true" editable="true" data-
type="string">New</Name>
</Status>
```

A list of all available Ticket status objects can be obtained by using a URI similar to the following:

```
https://my-servicedesk.parature.com/api/v1/4/52/Ticket/status?_token_=SOZh...
```

Ticket List By Status Type

When performing a query of Ticket objects via the List operation, a special query filter can be used to specify only Tickets with statuses that are in an "open" or "closed" state. The filter is a URI query string parameter named _status_type_, and the values for the status type are open and closed. The following example shows a List operation to retrieve only Tickets that are in an "open" status.

```
https://my-servicedesk.parature.com/api/v1/4/52/Ticket?_status_type_=open&_token_=SOZh...
```

A status is either "open" or "closed" according to the configuration of your service desk. Contact you Parature administrator for questions on specific statuses.

Ticket List For "My Tickets"

A specialized Ticket query filter exists that returns only the Tickets that are currently assigned to the CSR that is identified by the _token_parameter in the URI. This can be used as an alternative to looking up the CSR ID and specifying the ID as the _Assigned_To= value of the URI. For example, given a CSR whose ID is 12345 and _token_ value ismOomsTYzuavJHrgvhYVmGpwZQqGF2N5Uj8Dj9BMqtjqCWSGcFJP1@m/QFmzVakgo, the following two queries produce the same results:

```
https://my-servicedesk.parature.com/api/v1/4/52/Ticket?_Assigned_To_=12345&_token_=mOomsTYzuavJHrgvhYVmGpwZQqGF2N5Uj8Dj9BMqtjqCWSGcFJP1@m/QFmzVakgo
https://my-servicedesk.parature.com/api/v1/4/52/Ticket?_my_tickets_=true&_token_=mOomsTYzuavJHrgvhYVmGpwZQqGF2N5Uj8Dj9BMqtjqCWSGcFJP1@m/QFmzVakgo
```

The _my_tickets_=true query string parameter is provided as a convenience, since the Assigned_To field may not be readily available at the time of the query.

Parature API Download Object

First, read API Operations to understand how the API operations work for all objects. This page describes only those portions of the API that are specific to the Download object and assumes you are familiar with the default behavior.

Parature Download is composed of two objects. There is the Download *object*, which specifies characteristics of the Download, such as the display name, whether the Download is visible on the portal, the folder in which it exists, which SLAs can access it, et cetera. Additionally, for most Downloads there is also a physical file that is hosted within your Parature system. Alternately, it's possible to define a Download object for a file that is not hosted on the Parature system, using the External_Link value.

Steps for Each Operation

List, Retrieve and Schema

The list, retrieve and schema operations for Download follow the normal patterns outlined in the Operations section of the manual. There are two main differences in the Downloads XML object. See the example below.

1. The Download XML object contains a Globally Unique Identifier (GUID) that uniquely identifies the file object associated with this download, or an External_Link if the download file is not hosted at Parature.
2. The Download XML contains a reference to the Folder containing the Download.

The following request lists all the downloads in a specific folder, indicated by id value 26159.

```
https://mysandbox.parature.com/api/v1/4100/4549/Download?Folder_id_=26159&_token_=SOZhBs4hs...
```

Knowledge Base Article API

Read about API Operations to understand how the API operations work for all objects. This page describes only those portions of the API that are specific to the Knowledgebase Article object and assumes you are familiar with the common behavior.

List Behavior

In addition to the standard list behavior, the Knowledgebase (KB) module allows a keyword-based search. The keyword-based search cannot be combined with the regular field-specific behavior.

To search using keywords include the *keywords* query parameter. Note that you must encode spaces which appear in the keywords by replacing them with the plus sign, e.g.

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Article?_keywords_=Central+America&_tok
en_=ABC123XYZ...
```

This will return all KB articles whose question, alternate question, or answer contains ALL of the words indicated; both Central and America. You can also perform an ANY word search or exact match search by including the *searchoption*parameter. The options are: allwords, anywords, and exactwords. So in our example, to list all KB Articles whose questions or answers contain **either** Central or America use the URL:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Article?_keywords_=Central+America&_sea
rchoption_=anywords&_token_=ABC123XYZ...
```

The one field specification that can be used with the keyword search behavior is specifying the Folder or category to which the article belongs. To search only for articles that are within a specific folder an d its sub-folders specify the folder name using the Folders_id_ keyword. For example, to search all Articles in the Basic Questions folder, you would first obtain the folder id, possibly by using the list call, e.g.

```
https://myfarm-sandbox.parature.com/api/v1/33/420/ArticleFolder?_token_=5VJLevBgyr...
```

The list operation will return a list of folder objects, such as the following.

```
<ArticleFolder id="258" uid="33/420/ArticleFolder/258" href="https://myfarm-
sandbox.parature.com/api/v1/33/420/ArticleFolder/258">
<Name display-name="Name" required="true" editable="true" data-
type="string">Basic Questions</Name>
<Parent_Folder display-name="Parent Folder" required="true" editable="true"
data-type="entity">
<ArticleFolder id="1" uid="4/52/Articlefolder/1"
href="https://paradesk114/api/v1/4/52/Articlefolder/1">
<Name display-name="Name">Main Topic</Name>
</ArticleFolder>
</Parent_Folder>
</ArticleFolder>
```

To search all the Articles in the folder Basic Questions that contain both the words 'car insurance' submit the URL like:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Article?Folders_id_=258&_keywords_=car+
insurance&_searchoption_=allwords&_token_=ABC123XYZ...
```

NOTE (troubleshooting):* If the article search is not returning the results it should, please contact your Parature administrator to insure that your Service Desk search is indexed correctly.

Article Workflow and Create/Update API

The exceptions described in this section only apply if your Parature system is configured to have workflow enabled for Knowledgebase Articles. If you do not have workflow, ignore this section. If you are not sure if you have workflow or not, log into your Service Desk and browse to the Knowledgebase tab. If Browse by Status appears in the left pane, you have workflow enabled.

Article workflow is currently unavailable via the API. Thus, you cannot control the publication/workflow status of articles from the API. If you set the Published boolean value on a create or update call, it will be ignored. Articles created via the API will default to Draft status. An article can only be moved to Review or Published status from the Service Desk. Articles modified via the API will remain in their current status.

Article Folders

A ArticleFolder is a secondary object within the Parature system. Folders can be managed via the API, allowing all the normal list, retrieve, create, update and delete operations. A simple example is shown below.

The results of the list operation.

<Entities total="1" results="1" page="1" page-size="25" href="https://my-sandbox.parature.com/api/v1/4100/4549/Download">
<Download id="65296" uid="4100/4549/Download/65296" href="https://my-sandbox.parature.com/api/v1/4100/4549/Download/65296">
<Published display-name="Published" required="true" editable="true" data- type="boolean">true</Published>
<Visible display-name="Visible" required="false" editable="true" data- type="boolean">true</Visible>
<Description display-name="Description" required="false" editable="true" data- type="string">Calendar of events for October</Description>
<External_Link display-name="External Link" required="false" editable="true" data-type="string"/>
<Guid display-name="Attachment GUID" required="false" editable="true" data-type="string">2c1d96160a1a4157882e3bbf84f6fd13</Guid>
<Name display-name="Name" required="true" editable="true" data- type="string">OctoberCalendar</Name>
<Title display-name="Title" required="true" editable="true" data- type="string">OctoberCalendar</Title>
<Date_Created display-name="Date Created" required="false" editable="false" data-type="date">2007-11-06T21:13:08.470Z</Date_Created>
<Date_Updated display-name="Date Updated" required="false" editable="false" data-type="date">2007-11-06T21:13:08.470Z</Date_Updated>
<Folder display-name="Folder" required="true" editable="true" data-type="entity">
<DownloadFolder id="26159" uid="4100/4549/Downloadfolder/26159" href="https://my-sandbox.parature.com/api/v1/4100/4549/Downloadfolder/26159">
<Name display-name="Name">Product Instructions</Name>
</DownloadFolder>
</Folder>
<Permissions display-name="List of permissions" required="false" editable="true" data-type="entity" multi-value="true">
<Sla id="4748" uid="4100/4549/Sla/4748" href="https://my-sandbox.parature.com/api/v1/4100/4549/Sla/4748">
<Name display-name="Name">System Default</Name>
</Sla>
</Permissions>
</Download>
</Entities>

Create

The Create operation for a Download that uses an External_Link follows the normal pattern outlined in the create documentation. All required fields must be supplied, including a folder id, which will likely be obtained using a list operation on the DownloadFolder object.

To create a Download that references a file hosted at Parature there is a three-step process.

Step 1: Get an upload URI

A file upload URI must be obtained from the API. This upload URI in then used to upload the attachment file. The upload URI contains a temporary token that expires after a short amount of time; therefore, the upload URI should only be used for a single attachment operation. To obtain the upload URI, make a GET request to the Download/upload operation like the one shown below.

```
https://my-
sandbox.parature.com/api/v1/4100/4549/Download/upload?_token_=SOZhBs4hs7...
```

Upon success the call will return a XML response that includes an Upload tag like this:

```
<Upload href="http://my-
sandbox.parature.com/FileManagement/Upload/?token=Bp3sRV@8WCFBG...&amp;UploadID=501
550843"/>
```

Note that the href value includes an XML escaped ampersand character (&). This character must be un-escaped when using the upload URI so that the URI is as follows:

```
.../FileManagement/Upload/?token=Bp3sRV@8WCFBG...&UploadID=501550843
```

Step 2: Upload a file

Use the upload URI returned in the href attribute of the Upload element in step 1 to upload the file. Make a POST request to the URL indicated, with the file in the request body in the standard HTTP upload format (RFC 1867). See below is an example of the POST request body, that uploads a simple text file containing the words "This is my test".

```
-----------------------------123456789
Content-Disposition: form-data; name="MyTestFile"; filename="Testing.txt" Content-
Type: application/octet-stream

This is my test
-----------------------------123456789--
```

If the file uploads successfully you'll see XML results returned similar to the following

```
<result>
<passed>
<file>
<inputname>MyTestFile</inputname>
<filename>Testing.txt</filename>
<guid>d6f72e4460ac40e893a9340f72ee4fc1</guid>
</file>
</passed>
</result>
```

Step 3: Create the Download Object With the Newly Uploaded File

The XML returned from the successful file upload operation contains a guid tag. This Guid value is the unique identifier within your Parature system for the file you just uploaded. To create the new Download object, perform a create operation. In addition to supplying all the required fields you must supply the Guid of the file you just uploaded. This will cause the file to be associated with the download object.

Update

The update operation for Downloads follows all the standard behavior described in the update section of this manual with the exception of the file reference handling described below. Update requests can have a number of different actions with regard to the file attachment of the Download object, based on the state of the Download before the update, and the parameters supplied. In all cases only a GUID OR External_Link value should be supplied, not both.

Note, that if you are updating the Download with a new file, you must upload the file following steps 1 and 2 outlined for the Create process above. You then use the new GUID to update the Download.

Original Reference Type	Reference Supplied in Update	Resulting Behavior on Download Object
Guid	Same Guid as original	No change
Guid	Different Guid from original	Guid reference changed to new value, file referenced by old Guid physically deleted
Guid	External_Link	External_Link value set, original Guid reference removed,file referenced by old Guid physically deleted
External_Link	Same External_Link as original	No change
External_Link	Different External_Link than original	External_Link set to new value
External_Link	Guid	External_Link value removed, Guid reference added

Delete

Use the normal deletion operation defined in the operations section of this manual and both the Download object, and the associated file will be deleted.

*Note: Deletion of Download items results in a "virtual" deletion of the Download. After performing the delete operation the Download will appear in the trash bin on the service desk. If you want the Download
to be physically removed from the system immediately you can use the _purge_=true option on the delete operation. Purged Downloads cannot be recovered.

Download Folders

A DownloadFolder is a secondary object within the Parature system. Folders can be managed via the API, allowing all the normal list, retrieve, create, update and delete operations. A simple example is shown below.

List Download Folders

The DownloadFolder object type is used to retrieve a list of all folders within the Download module.

```
https://my-sandbox.parature.com/api/v1/4100/4549/DownloadFolder?_token_=SOZhBs...
<Entities total="1" results="1" page="1" page-size="25" href="https://my-
sandbox.parature.com/api/v1/4/52/DownloadFolder">
<DownloadFolder id="194330" uid="4100/4549/DownloadFolder/19430"
href="https://my- sandbox.parature.com/api/v1/4100/4549/DownloadFolder/19430">
<Is_Private display-name="Is Private" required="false" editable="true" data-
type="boolean">false</Is_Private>
<Description display-name="Description" required="false" editable="true" data-
type="string"/>
<Name display-name="Name" required="true" editable="true" data- type="string">Top
Level Folder</Name>
<Date_Updated display-name="Date Updated" required="false" editable="false" data-
type="date">2006-07-10T17:14:09.220Z</Date_Updated>
<Parent_Folder display-name="Parent Folder" required="true" editable="true" data-
type="entity">
<DownloadFolder id="1022" uid="4100/4549/DownloadFolder/1022" href="https://my-
sandbox.parature.com/api/v1/4100/4549/DownloadFolder/1022">
<Name display-name="Name">My Downloads</Name>
</DownloadFolder>
</Parent_Folder>
</DownloadFolder>

...
<Entities>
```

Retrieving a Download Folder

To retrieve the details of a folder object, the DownloadFolder object type and ID are used in the URI of the retrieve/GET command as follows:

```
https://my-
sandbox.parature.com/api/v1/4100/4549/DownloadFolder/8675309?_token_=SOZhBs...
```

Creating a Download Folder

A DownloadFolder can be created using a create/POST command.

```
https://my-sandbox.parature.com/api/v1/4100/4549/DownloadFolder?_token_=SOZhBs...
```

The request body must contain the required information for creating the DownloadFolder.

```
<DownloadFolder>
<Is_Private>false</Is_Private>
<Description>My new folder for testing</Description>
<Name>ETest</Name>
<Parent_Folder>
<DownloadFolder id="19430" \>
</Parent_Folder>
</DownloadFolder>
```

Updating a Download Folder

A DownloadFolder can be updated using an update/PUT command.

```
https://my-
sandbox.parature.com/api/v1/4100/4549/DownloadFolder/12345?_token_=SOZhBs...
```

The request body must contain all information for the DownloadFolder. Any optional fields that are not submitted in the body are set to null/blank values.

```
<DownloadFolder>
<Is_Private>true</Is_Private>
<Description>My new folder for testing - updated the description</Description>
<Name>ETest</Name>
<Parent_Folder>
<DownloadFolder id="19430" \>
</Parent_Folder>
</DownloadFolder>
```

Deleting a Download Folder

Unlike most of the first order obejcts, such as Customers, Tickets, and Articles, folders do not have a "trashed" state from which they can be purged or untrashed. Folders can only be purged. However, before the API purges a folder, a check is performed to determine if the folder is empty. Only empty folders can be purged. Attempting to purge a folder that contains any objects or subfolders will result in an error. To purge a DownloadFolder, the Delete operations is performed using the following URI:

```
https://my-
sandbox.parature.com/api/v1/4100/4549/DownloadFolder?_purge_=true&_token_=SOZhBs...
```

Parature API - Account Example

This example will demonstrate all the steps of updating an Account object. This Account module has the built-in system fields, plus six custom fields.

Static/System Fields:

- Account Name - is a system field containing the name of the account. It is of type string.
- Owned By - is a system field containing the CSR that created the account. It is of type entity, as it indicates a CSR entity/object.
- SLA - is a system field containing the Service Level Agreement associated with the account. It is of type entity.
- Modified By- is a system field containing the CSR that last modified the account. It is of type entity.

Custom Fields:

- Partner Type - This is of type option, and does not allow multi-select.
- Certified Trainer - This custom field is of type checkbox, which returns a boolean data type.
- PLB Specialization - Is a multi-select option/dropdown field.
- Partner Description - Is a text field.
- Enrollment Date - is a date field.
- Certified Professionals - Is an integer field.

Schema

Below is an example of what a schema request would return for this Account schema request. By default, the schema operation does not return a standard schema format (such as XSD or RelaxNG), but instead returns a template object, showing all the fields in the object. https://myfarm-sandbox.parature.net/api/v1/4/52/Account/schema?_token_=blVPu0hK6T6MiMX....

```
<Account>
<Account_Name display-name="Account Name" required="true" editable="true" data-
type="string" />
<Date_Created display-name="Date Created" required="false" editable="false" data-
type="date"/>
<Date_Updated display-name="Date Modified" required="false" editable="false"
data-type="date"/>
<Default_Customer_Role display-name="Default Customer Role" required="false"
editable="true" data-type="entity">
<CustomerRole>
<Name display-name="Name"/>
</CustomerRole>
</Default_Customer_Role>
<Modified_By display-name="Modified By" required="false" editable="false" data-
type="entity">
<Csr>
<Full_Name display-name="Full Name" />
</Csr>
</Modified_By>
<Owned_By display-name="Owned By" required="false" editable="false" data-
type="entity">
<Csr>
<Full_Name display-name="Full Name" />
```

```
</Csr>
</Owned_By>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla>
<Name display-name="Name" />
</Sla>
</Sla>
<Custom_Field id="87848" display-name="Certified Trainer" required="false"
editable="true" data-type="boolean" />
<Custom_Field id="87849" display-name="Partner Type" required="true"
editable="true" data-type="option" multi-value="false">
<Option id="58">
<Value>OEM Partner</Value>
</Option>
<Option id="59">
<Value>Reseller</Value>
</Option>
</Custom_Field>
<Custom_Field id="87850" display-name="PLB Specialization" required="false"
editable="true" data-type="option" multi-value="true">
<Option id="7">
<Value>Product A</Value>
</Option>
<Option id="8">
<Value>Product B</Value>
</Option>
<Option id="9">
<Value>Product C</Value>
</Option>
<Option id="60">
<Value>All</Value>
</Option>
</Custom_Field>
<Custom_Field id="87851" display-name="Partner Description" required="false"
editable="true" data-type="string" />
<Custom_Field id="87852" display-name="Enrollment Date" required="true"
editable="true" data-type="datetime" />
<Custom_Field id="87853" display-name="Certified Professionals" required="false"
editable="true" data-type="int" />
</Account></Account>
```

List

For our example, we will assume we want to update an existing account object. First, we will call the list operation to get a list of the Account objects. The return is shown below, and has been reduced to 2 objects for the sake of brevity. https://myfarm-sandbox.parature.net/api/v1/4/52/Account?_token_=blVPu0hK6T6MiMX....

```
<Entities total="2" results="2" page="1" page-size="25" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Account">
<Account id="6" uid="4/52/Account/6" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Account/6" service-desk- uri="https://myfarm-
sandbox.parature.com/ics/am/amDetail.asp?amID=6">
<Date_Created display-name="Date Created" required="false" editable="false"
type="date">2007-09-22T22:17:50.593</Date_Created>
<Date_Updated display-name="Date Updated" required="false" editable="false"
type="date">2007-09-22T22:17:50.593</Date_Updated>
<Default_Customer_Role display-name="Default Customer Role" required="false"
editable="true" data-type="entity">
<CustomerRole id="3933" uid="4139/4549/Customer/role/3933"
href="https://qa1.parature.net/api/v1/4139/4549/Customer/role/3933">
<Name display-name="Name">Account Administrator</Name>
</CustomerRole>
</Default_Customer_Role>
<Account_Name display-name="Account Name" required="true" editable="true" data-
type="string">Calvertron</Account_Name>
<Owned_By display-name="Owned By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Test User</Full_Name>
</Csr>
</Owned_By>
<Modified_By display-name="Modified By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Test User</Full_Name>
</Csr>
</Modified_By>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="10" uid="4/52/Sla/10" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Sla/10">
<Name display-name="Name">Registered Enterprise Company</Name>
</Sla>
</Sla>
</Account>
<Account id="7" uid="4/52/Account/7"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Account/7" service-desk-
uri="https://myfarm-sandbox.parature.com/ics/am/amDetail.asp?amID=6">
<Date_Created display-name="Date Created" required="false" editable="false" data-
type="date">2008-06-09T20:51:12.300Z</Date_Created>
<Date_Updated display-name="Date Modified" required="false" editable="false" data-
type="date">2008-07-20T08:42:21.507Z</Date_Updated>
<Default_Customer_Role display-name="Default Customer Role" required="false"
editable="true" data-type="entity">
<CustomerRole id="3933" uid="4139/4549/Customer/role/3933"
href="https://qa1.parature.net/api/v1/4139/4549/Customer/role/3933">
```

```
<Name display-name="Name">Customer</Name>
</CustomerRole>
</Default_Customer_Role>
<Account_Name display-name="Account Name" required="true" editable="true" data-
type="string">Mega Network</Account_Name>
<Owned_By display-name="Owned By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Test User</Full_Name>
</Csr>
</Owned_By>
<Modified_By display-name="Modified By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Test User</Full_Name>
</Csr>
</Modified_By>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="7" uid="4/52/Sla/7" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Sla/7">
<Name display-name="Name">System Default</Name>
</Sla>
</Sla>
</Account>
</Entities>
```

Retrieve

We will now retrieve details for the Calvertron account. https://myfarm-sandbox.parature.net/api/v1/4/52/Account/6?_token_=blVPu0hK6T6MiMX ... This returns the following results.

```
<Account id="6" uid="4/52/Account/6" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Account/6" service-desk- uri="https://myfarm-
sandbox.parature.com/ics/am/amDetail.asp?amID=6">
<Date_Created display-name="Date Created" required="false" editable="false"
type="date">2007-09-22T22:17:50.593</Date_Created>
<Date_Updated display-name="Date Updated" required="false" editable="false"
type="date">2007-09-22T22:17:50.593</Date_Updated>
<Default_Customer_Role display-name="Default Customer Role" required="false"
editable="true" data-type="entity">
<CustomerRole id="3933" uid="4139/4549/Customer/role/3933"
href="https://qa1.parature.net/api/v1/4139/4549/Customer/role/3933">
<Name display-name="Name">Account Administrator</Name>
</CustomerRole>
</Default_Customer_Role>
<Account_Name display-name="Account Name" required="true" editable="true" data-
type="string">Calvertron</Account_Name>
<Owned_By display-name="Owned By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Fred Flintstone</Full_Name>
</Csr>
</Owned_By>
<Modified_By display-name="Modified By" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">Fred Flintstone</Full_Name>
</Csr>
</Modified_By>
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="10" uid="4/52/Sla/10"
href="https://myfarm-sandbox.parature.com/api/v1/4/52/Sla/10">
<Name display-name="Name">Registered Enterprise Company</Name>
</Sla>
</Sla>
<Custom_Field id="87849" display-name="Partner Type" required="true" editable="true"
data-type="option" multi-value="false">
<Option id="58" selected="true">
<Value>OEM Partner</Value>
</Option>
<Option id="59">
<Value>Reseller</Value>
</Option>
</Custom_Field>
<Custom_Field id="87848" display-name="Certified Trainer" required="false"
editable="true" data-type="boolean">1</Custom_Field>
<Custom_Field id="87851" display-name="Partner Description" required="false"
editable="true" data-type="string">Calvertron manufactures and sells widgets in
additional to providing consulting service for their own line of widgets and our full
line of gadgets and products.</Custom_Field>
```

```
<Custom_Field id="87853" display-name="Certified Professionals" required="false"
editable="true" data-type="int">12</Custom_Field>
<Custom_Field id="87850" display-name="PLB Specialization" required="false"
editable="true" data-type="option" multi-value="true">
<Option id="7" selected="true">
<Value>Product A</Value>
</Option>
<Option id="8" selected="true">
<Value>Product B</Value>
</Option>
<Option id="9">
<Value>Product C</Value>
</Option>
<Option id="60">
<Value>All</Value>
</Option>
</Custom_Field>
<Custom_Field id="87852" display-name="Enrollment Date" required="true"
editable="true" data-type="datetime">2007-06-06T00:00:00</Custom_Field>
</Account>
```

Update

In this example, we will update the Calvertron client (id=6). We will change the SLA from Registered Enterprise Company to System Default, and we will change the Partner Type from OEM Partner to Reseller.

The Partner Type field contains a list of string values as options. All the possible values for the field are returned in the details for the object.

The SLA field is an entity type and points to another business object as the value for the field. To find the possible values for this field we must first look up the available objects for this entity type. As we can see from the Sla field definition, the object type it points to is Sla. In the snippet below, the outer Sla tag is for the SLA field inside the Account object, and the inner Sla tag refers to an object of type Sla, to which the field points. See Object Types for a listing of possible object types. Notice that the Sla entity reference includes an XML Object LocationURI.

```
<Sla display-name="Service Level Agreement" required="true" editable="true" data-
type="entity">
<Sla id="10" uid="4/52/Sla/10" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Sla/10">
<Name display-name="Name">Registered Enterprise Company</Name>
</Sla>
</Sla>
```

List of SLAs

To obtain the list of SLAs we make the following request: https://myfarm-sandbox.parature.net/api/v1/4/52/Sla?_token_=blVPu0hK6T6MiMX and the following list is returned.

```
<Entities total="2" results="2" page="1" page-size="25" href="https://myfarm-
sandbox.parature.net/api/v1/4/52/Sla">
<Sla id="7" uid="4/52/Sla/7"" href="https://myfarm-
sandbox.parature.net/api/v1/4/52/Sla/7">
<Name display-name="Name" required="true" editable="false" data- type="string">System
Default</Name>
</Sla>
<Sla id="10" uid="4/52/Sla/10"
href="https://myfarm-sandbox.parature.net/api/v1/4/52/Sla/10">
<Name display-name="Name" required="true" editable="false" data-
type="string">Registered Enterprise Company</Name>
</Sla>
</Entities>
```

Updating the Record

We want to change the Calvertron SLA to System Default, so we will use id=7 from the list provided in the retrieve response. We also want to change the Partner Type to Reseller, so we will use id=59 from the original retrieve request on Calvertron. Note that in the update call below that we re-submit all of the Calvertron fields, except those marked as editable=false. We resubmit all fields, because submitting an update without the fields is the equivalent of directing the Update API to delete those field values. However, we omit the non-editable fields because they cannot be changed via the API and would cause validation errors.

```
<Account id="6">
<Account_Name>Calvertron</Account_Name>
<Sla>
<Sla id="7" />
</Sla>
<Custom_Field id="87849">
<Option id="59" selected="true" />
</Custom_Field>
<Custom_Field id="87848">1</Custom_Field>
<Custom_Field id="87851">Calvertron manufactures and sells widgets in additional to
providing consulting service for their own line of widgets and our full line of
gadgets and products.</Custom_Field>
<Custom_Field id="87853">12</Custom_Field>
<Custom_Field id="87850">
<Option id="7"    selected="true" />
<Option id="8"    selected="true" />
</Custom_Field>
<Custom_Field id="87852">2007-06-06T00:00:00</Custom_Field>
</Account>
```

Default Customer Role field

The **Default Customer Role** field is an entity field that determines what role is assigned to new Customers in an Account. Example:

```
<Account>
...
<Default_Customer_Role display-name="Default Customer Role" required="false"
editable="true" data-type="entity">
<CustomerRole id="3252" uid="4/52/Customer/role/3252" href="https://myfarm-
sandbox.parature.com/api/v1/4/52/Customer/role/3252">
<Name display-name="Name">Customer (Limited)</Name>
</CustomerRole>
</Default_Customer_Role>
...
</Account>
```

If a new Customer was created within this Account, or an existing Customer was edited and associated with this account, they would inherit this Default Customer Role. If a specific Role is given to the Customer as part of the creation or edit, the Customer will receive that Role, instead of the Default Customer Role. Searching and Sorting by this field is available. For more information on how to filter or sort lists, read about the List Operation.

Default Role Value

If an account is created via the API and the Default_Customer_Role is not specified, the default customer role for the Account will be set to "Customer".

Service Desk URI

A new entity attribute is available, called service-desk-uri. This is the URL you can use to navigate to the entity in the Service Desk. The attribute will show up on the main entity being retrieved, but not on the "reference fields" as shown in the example below:

```
<Ticket id="5" uid="4/52/Ticket/5" href="http://myfarm-sandbox.parature.com/api/v1/4/52/Ticket/5"
service-desk-uri="https://myfarm-
sandbox.parature.com/ics/tt/ticketDetail.asp?ticket_id=5">
...
<Assigned_To display-name="Assigned To" required="false" editable="false" data-
type="entity">
<Csr id="16" uid="4/52/Csr/16"
href="http://myfarm-sandbox.parature.com/api/v1/4/52/Csr/16">
<Full_Name display-name="Full Name">John Doe</Full_Name>
</Csr>
</Assigned_To>
...
</Ticket>
```

The following entities now have this attribute:

- Account
- Article
- Asset
- Customer
- Download
- Product
- Ticket

Note: The actual URL is subject to change at any time, we don't recommend saving these URL's in another system. This URL is the same as displayed for the entity when using the RSS output option.

Discussion - Modules

Knowledge Base

Question and Answer

Both the question and answer may contain HTML, which would qualify as pattern **seven**. However, all HTML present in the question and answer text should be HTML encoded when added to the XML document and consequently match pattern **four**.

Folders

An article must reside in one or more folders. Since folders are labeled as multi-value elements, pattern **six** will be matched

XML

```
<Folders>
<ArticleFolder id=6 >
<name>Folder Six</name>
</ArticleFolder>
<ArticleFolder id=12 >
<name>Folder Twelve</name>
</ArticleFolder>
</Folders>
```

JSON

```
Folders : {
ArticleFolder : [
{
"@id" : 6,
"name" : "Folder Six",
...
},
{
"@id" : 12,
"name" : "Folder Twelve",
...
}
]
}
```

Permissions (SLA and Products)

An article may grant permission to zero or more SLA groups. Similarly, an article may grant permission to zero or more products. The following example lists the JSON snippet for a few different cases involving SLA permissions. Both elements are multi-value and match pattern **six**

XML

```
//One SLA granted
<Permissions>
<SLA id=6>
<name>System Default SLA</name>
</SLA>
</Permissions>
```

JSON

```
//One SLA granted
Permissions : {
SLA : [{
"@id" : 6,
"name" : "System Default SLA",
...
}]
}
```

Download

Permissions

Please see the discussion in the Knowledge Base section

Accounts

Viewable Accounts

The account module supplies the system field, Viewable Accounts, which may contain zero or more references to other accounts and is multi-value matching pattern **six**

XML

```
//Two SLAs granted
<Shown_Accounts>
<Account id=6>
<name>System Default SLA</name>
</Account>
<Account id=12>
<name>Parature</name>
</Account>
</Shown_Accounts>
```

JSON

```
//Two viewable accounts
Shown_Accounts : {
Account : [
{
"@id" : 6,
"name" : "Softlogic",
...
},
{
"@id" : 12,
"name" : "Parature",
...
}
]
}
```

Discussion - Custom Fields

All custom fields are listed as Custom_Field with the id attribute differentiating between fields. This naming matches pattern **six**. This means that the custom field values of an object will be listed as an array. Specifics to various custom field types are discussed in the sub-sections below.

```
//XML
<Account>
<Custom_Field>
</Custom_Field>
<Custom_Field>
</Custom_Field>
</Account>

//JSON Account : {
Custom_Field : [
{ /*first custom field*/ },
{ /*second custom field*/ },
...
]
}
```

Non-select Fields

All non-select fields (text, password, date, etc.), will be displayed matching pattern **four**

```
//XML
<Custom_Field id=1234>
custom field value
</Custom_Field>
//JSON Custom_Field : {
"@id" : 1234,
"#text" : "custom field value"
```

Select Fields

All select type fields (dropdown, multi-dropdown, radio, etc.) will be displayed matching pattern **six**. This is true because all selectable options are always displayed for these types of fields.

```
//XML
<Custom_Field id=1234>
<option id=9876>
<value>Hello!</value>
</option>
<option id=9875 selected=true>
<value>Goodbye.</value>
</option>
</Custom_Field>

//JSON Custom_Field : {
"@id" : 1234, "option" : [
{
},
{
},
...
]
}
"@id" : 9876, "value" : "Hello!"
"@id" : 9875, "@selected" : true, "value" : "Goodbye."
```

ViewOrder

Each option will be returned with a viewOrder attribute, this is the order that the field's options are displayed in the Service Desk and Portal."

```
<Custom_Field id="24" display-name="Ticket Origin" required="false" editable="true"
data-type="option" multi-value="false">
<Option id="4" viewOrder="2">
<Value>Email</Value>
</Option>
<Option id="3" viewOrder="4">
<Value>Other</Value>
</Option>
<Option id="2" viewOrder="3">
<Value>Phone Call</Value>
</Option>
<Option id="1" viewOrder="1">
<Value>Web Site</Value>
</Option>
<Option id="19" viewOrder="5">
<Value>Facebook Wall</Value>
</Option>
<Option id="20" viewOrder="6">
<Value>Twitter</Value>
</Option>
<Option id="27" viewOrder="7">
<Value>Facebook Support</Value>
</Option>
</Custom_Field>
```

Alternate Output Formats

The default output format for all operations is XML. However, it is possible to specify an alternative output format by providing a value for the _output_ query parameter in the resource URI. These options are discussed below.

RSS

URI	Supported Operations
output=rss	List

The **List** and **Retrieve** operations can also produce output in Really Simple Syndication (RSS) format. The RSS output includes basic HTML formatting for each result in the result set. The RSS title will be the object's attributes, while the RSS description will contain an HTML formatted list of the object's direct child properties.

RSS Example

The following example shows a List operation that retrieves all tickets created on or after August 5, 2007, with the output in RSS format:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?_output_=rss&Date_Created_min_
=2007-08-05&_token_=ABC123XYZ...
```

JSON

URI	Supported Operations
output=json	List, Retrieve, Schema

Javascript Object Notation (JSON) is useful when developing javascript applications for the browser.

JSON Example

The following example shows a List operation that retrieves all tickets created on or after August 5, 2007, with the output in JSON format:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?_output_=json&Date_Created_min
_=2007-08-05&_token_=ABC123XYZ...
```

parature

JSON Introduction

JSON is a language independent data exchange format based on the object notation of the JavaScript lanaguage. Although based on JavaScript, reading and writing of JSON does not require JavaScript. In fact, reading and writing of JSON is currently support by most languages and frameworks. For more information about JSON itself, please visit the JSON homepage or wikipedia.

The Parature API supports JSON responses for the List, Retrieve and Schema operation types.

Basic JSON

A JSON object at its simplest level is an associative hash table, in which each item is defined as a key value pair. A JSON key may be any string value. A JSON primitive value may consist of any of the following:

- String Strings are surrounded by double quotes and contain Unicode characters or common backslash escapes
- Number A number is defined in the conventional sense. e.g. 1, 23405 or =-36
- Boolean A boolean value is either true or false
- null

Please note that Date is not included in this list. There is currently no standard for dates in JSON.
JSON object values may also include arrays. A JSON Array begins and ends with braces and contains comma delimited values enclosed in square braces. e.g. ["value1", "value2"]
Finally, an object itself is defined between curly braces, {}, in which members are defined as key value pairs separated with colons, :. Subsequent members are separated by commas, ,. All keys are specified as strings, with their values being any of the mentioned value types. The following defines a very simple JSON object

```
{
"title" : "Learning JSON",
"price" : {
"currency" : "dollars",
"amount" : 23.99
},
"pages" : 236,
"copyright" : true,
"comments" : [ "Good book.",  "Boring." ]
}

//Assuming this value is returned from the API in the variable named "result", values
would be accessed as such
var title = result.title;
var priceAmount = result.price.amount;
var firstComment = result.comments[0];
//Or using the associative hash table aspect of JSON
var title = result["title"];
var priceAmount = result["price"]["amount"];
var firstComment = result["comments"][0];
```

JSON output in Parature API

JSON is supported for the following operations on all entities:

- List
- Retrieve
- Schema

JSON output may be requested from the Parature API by specifying _output_=json in the resource URI. JSON responses will have content type application/json.

Example

The following example shows a List operation that retrieves all tickets created on or after August 5, 2007, with the output in JSON format:

```
https://myfarm-
sandbox.parature.com/api/v1/33/420/Ticket?_output_=json&Date_Created_min_=2007
-08-05&_token_=ABC123XYZ...
```

Results from this request could be used in the following manner,

```
/*
*     Call this method on the result of a successful XmlHttpRequest to the API.
*/
function onXmlHttpSuccess(/*string*/ apiJsonResult) {
var result = eval("(" + apiJsonResult + ")");

//interact with "result" just like any other javascript object
}
```

Please note that it is necessary to surround the JSON text within parentheses. This applies closure to the eval and consequently returns the value into the result variable.

JSON Callback support

Since JSON is native JavaScript, it is possible to load the Parature API response as active JavaScript in which a method in the client page may be invoked. This process is typically referred to as JSON-P and may be accomplished in the following manner,

1. Specify _output_=javascript
2. Specify the callback method in the URI. Use the URL parameter _callback_

Please note that the output must be specified as javascript and not json since the response generated will actually be a javascript document invoking a javascript method in the client DOM.

The _callback_ parameter is used to specify the name of a method in the client to be invoked with input equal the JSON representation of the resource. Callback function names may use only upper and lowercase alphabetic characters (A-Z, a-z), numbers (0-9), and the underscore (_). Requests that fail to meet this requirement will result in a response of a BadApiUriException.

JSON and callbacks are very useful when dealing in client-side JavaScript. Using JSON with callbacks, a request to a Parature API resource may be loaded as the source of an HTML script tag and use the result of the resource request directly elsewhere in the client side page. All of this may be accomplished free of the security restrictions headache involved with XmlHttpRequest interaction such as using proxies or server rewrites.

Please Note

The use of the _callback_ functionality as described above exposes the entire DOM of the invoking client- side JavaScript site to the Parature API response. Although Parature API responses contain only the output described above (the requested resource data as a parameter to the_callback_ function invocation), security minded developers may want to use the conventional AJAX web service methods.

parature

XML to JSON Design

Summary

The XML to JSON conversion used in the Parature API is based on the specifications defined by Stefan Goessner in his article published on xml.com. In this article, he defines seven patterns in which an XML element may be defined. Instances of each of these seven patterns may be identified in some instances of Parature API responses. These instances are discussed below.

Patterns

Pattern	XML	JSON	Access
1	<e/>	"e": null	o.e
2	<e>text</e>	"e": "text"	o.e
3	<e name="value" />	"e":{"@name": "value"}	o.e["@name"]

4	<e name="value">text</e>	"e": {
		"@name": "value", "#text":
		"text"
		}
			o.e["@name"]
			o.e["#text"]

5	<e> <a>text
	text</e>
		"e": {
		"a": "text", "b": "text"
		}
			o.e.a o.e.b

6	<e> <a>text
	<a>text</e>
		"e": {
		"a": ["text", "text"]
		}
			o.e.a[0]
			o.e.a[1]

7	<e> text <a>text</e>	"e": {
		"#text": "text", "a": "text"
		}
			o.e["#text"]
			o.e.a

Special Value Types

The following action is taken when the following situations are encountered,

- CDATA values will equate to a property named "@cdata-text" with a value equal to the value of the unformatted CDATA value
- reserved escape data will be preserved in its same escape sequence (e.g. \')
- HTML values will be treated like unformatted CDATA, with the property name equal to that of XML node (see "Semi-Structured XML" in the article referenced above)
- Dates -- No standard exists for dates and they will be processed as is.

RSS Feeds

RSS stands for Really Simple Syndication, a type of Web feed format used to publish frequently updated content in a compact format. The Parature API allows clients to feed data out of their system in RSS 2.0 format. These RSS feeds can be used in a variety of ways.

For example, an RSS feed of new tickets or knowledge base content could be created for internal use, making it simple to keep track of new Service Desk information from any RSS-enabled browser or reader. We **do not recommend** using RSS feeds externally, as they contain API tokens and would be a security risk.

Please contact your Account Executive if you are interested in adding RSS capabilities to your Parature support solution

Special Cases

Various elements throughout the Parature schema allow the association of zero or more sub-elements. This quality of allowing, but not requiring, multiple values creates the opportunity for inconsistency between property access in the JSON output. To elaborate, please consider the following example involving Permissions granted to a given record. This example shows the conversion of a Permissions node that contains (a) zero, (b) one, and (c) two sub-nodes. The XML cases are listed first, with the corresponding JSON afterwards.

```
/*    XML
*/
//No permission granted (zero)
<Permissions />
//One SLA granted
<Permissions>
<SLA id=6>
<name>System Default SLA</name>
</SLA>
</Permissions>
//Two SLAs granted
<Permissions>
<SLA id=6>
<name>System Default SLA</name>
</SLA>
<SLA id=12>
<name>Platinum</name>
</SLA>
</Permissions>

/*    JSON
*/
//No permission granted
Permissions : null
//One SLA granted
Permissions : {
SLA : {
"@id" : 6,
"name" : "System Default SLA",
...
}
}
//Two SLAs granted
Permissions : { SLA : [
{
"@id" : 6,
"name" : "System Default SLA",
},
{
"@id" : 12,
"name" : "Platinum",
...
}
]
}
```

Although it is possible for an API consumer to programmatically compare the Permissions property against null and subsequently its type against Array, such a model would be cumbersome. For this reason, the Parature API's XML to JSON conversion takes an extra step when generating the JSON.

Any element that is marked with the multi-value="true" attribute will consistently represent its values as an array regardless of their cardinal number. Such a truism holds that the previous example would actually appear as such,

```
//No permission granted
Permissions : { SLA : []
}
//One SLA granted
Permissions : { SLA : [{
"@id" : 6,
"name" : "System Default SLA",
...
}]
}
//Two SLAs granted
Permissions : {
SLA : [
{
"@id" : 6,
"name" : "System Default SLA",
...
},
{
"@id" : 12,
"name" : "Platinum",
...
}
]
}
```

Array Element Key Names

Unfortunately the qualification of multi-value does not provide a complete set of those elements that contain repeatable sub-elements or are themselves repeatable (e.g. Custom_Field). For this reason, the following elements will be represented as arrays based on their own name or their parent-node's name alone:

Element Type	Element Names
Node itself	Custom_Field Option
Children of these nodes	Entities Products Ticket_Children Ticket_Attachments Actions ActionHistory History_Attachments