TAXII™ 2.0 Specification
Version 2.0-draft-3

Technical Committee
OASIS Cyber Threat Intelligence (CTI) TC

Chair
Richard Struse (Richard.Struse@hq.dhs.gov), DHS Office of Cybersecurity and Communications (CS&C)

Editors
Bret Jordan (bret.jordan@bluecoat.com), Blue Coat Systems, Inc.
Mark Davidson (mdavidson@soltra.com), Soltra

Document Table of Contents

1. Introduction
   1.1. Terminology
   1.2. Overview
      1.2.1. Channels Overview
      1.2.2. Collections Overview
   1.3. Document Conventions
      1.3.1. Naming Conventions
      1.3.2. Font Colors and Style

2. HTTPS Requirements

3. DNS SRV Records
   3.1. Requirements
   3.2. Example

4. Discovery API
   4.1. Requirements
   4.2. Examples
5. API Root
   5.1. Requirements
   5.2. Examples

6. Content Negotiation
   6.1. TAXII Media Type
       6.1.1. Requirements
   6.2. STIX Media Type
       6.2.1. Requirements

7. Primitive Types

8. TAXII API
   8.1. URL Endpoint Summary
   8.2. URL Parameters
   8.3. Global Requirements
       8.3.1. Property and String Requirements
       8.3.2. Server Requirements
       8.3.3. Client Requirements
   8.4. HTTP Status Codes
   8.5. GET <discovery>
       8.5.1. Requirements
       8.5.2. Examples
   8.6. GET <api-root>
       8.6.1. Requirements
       8.6.2. Examples
   8.7. GET /collections
       8.7.1. Requirements
       8.7.2. Examples
   8.8. GET /collections/<name>
       8.8.1. Requirements
       8.8.2. Examples
   8.9. GET /collections/<name>/manifest
       8.9.1. Requirements
       8.9.2. Examples
   8.10. GET /collections/<name>/objects
       8.10.1. Requirements
       8.10.2. Examples
   8.11. POST /collections/<name>/objects
       8.11.1. Requirements
8.12. GET /collections/<name>/objects/<object-id>
  8.12.1. Requirements
  8.12.2. Examples
8.13. GET /object-search
  8.13.1. Requirements
  8.13.2. Examples
8.14. GET /status/<status-id>
  8.14.1. Requirements
  8.14.2. Examples

9. TAXII Resources
  9.1. API Root Resource
  9.2. Discovery Resource
  9.3. Collection Resource
  9.4. Error Resource
  9.4.1. Example
  9.5. Manifest Resource
  9.6. Object Resource
  9.7. Status Resource

10. Customizing TAXII Resources
  10.1. Custom Properties
  10.1.1. Requirements

11. Conformance
  11.1. TAXII Servers
  11.2. Mandatory Features
    11.2.1. TODO
  11.3. Optional Features
    11.3.1. TODO

12. Appendix A. Acknowledgments
13. Appendix B. Changes from TAXII 1.1
14. Appendix C - End to End Workflow
15. Appendix D - Security Considerations
1. Introduction

TAXII is an open application layer protocol for the communication of cyber threat information. Focusing on simplicity and scalability, TAXII enables authenticated and secure communication of cyber threat information across products and organizations.

This specification defines the TAXII RESTful API and its resources along with the requirements for TAXII Client and Server implementations.

1.1. Terminology

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119 (REF:RFC2119).

API Root - An instance of the TAXII API that is often used to align to trust groups.
API Root URL - Is the “root” URL for any particular instance of the TAXII API.
Channel - A publish-subscribe communications method where messages are exchanged.
CTI - Cyber Threat Intelligence
Collection - Is a logical group of CTI objects.
Consumer - Any entity that receives content via TAXII.
Discovery Resource - Contains information about a TAXII Server, including the various API Roots that it knows about.
Message - A resource transmitted over a Channel from producer to consumer.
MTI - Mandatory to Implement
Object Resource - CTI that is often represented in STIX.
Producer - Any entity that sends content via TAXII.
Status Resource - Contains information about an action that was processed asynchronously.
STIX - Structured Threat Information Expression
STIX Content - STIX documents, including STIX Objects, grouped as STIX Bundles.
STIX Object - A STIX Domain Object (SDO) or STIX Relationship Object (SRO)
TAXII - Trusted Automated Exchange of Indicator Information
TAXII API -
TAXII Client -
TAXII Server - A system that supports the exchange of CTI between TAXII Clients known as producers and consumers.
1.2. Overview

This specification defines two communication methods - Channels and Collections - for transmitting CTI. TAXII is designed around a RESTful architecture and requires HTTPS-only communications (i.e., HTTP communications are not conformant with this specification).

Figure 1.1

1.2.1. Channels Overview

A Channel enables producers and consumers to exchange information in an asynchronous, event-based manner where both producers and consumers are TAXII clients of a TAXII server and the Channel is maintained by the TAXII server. A TAXII Server may contain multiple Channels per API Root and Channels are used to exchange information in a publish–subscribe manner.

Figure 1 below illustrates how Channel communications are used when a single producer sends a message to the TAXII Server, and that TAXII Server then distributes the message to all authorized consumers that have previously registered with the TAXII server. Normative requirements for Channels are defined later in this document.

1.2.2. Collections Overview

A Collection is used by TAXII clients to send information to the TAXII server or request information from the TAXII server. A TAXII Server may support multiple Collections per API Root, and Collections are used to exchange information in a request–response manner.
Figure 2 below illustrates how Collection based communications are used when a single TAXII Client makes a request to a TAXII Server and the TAXII Server fulfills that request with information available to the TAXII Server (nominally from a database). Normative requirements for Collections are defined later in this document.

1.3. Document Conventions

1.3.1. Naming Conventions

All type names, property names and literals are in lowercase. Words in property names are separated with an underscore (_), while words in type names and string enumerations are separated with a dash (-). All type names, property names, object names, and vocabulary terms are between three and 250 characters long.

1.3.2. Font Colors and Style

The following color, font and font style conventions are used in this document:

- The Consolas font is used for all type names, property names and literals.
  - resource and type names are in red with a light red background – collection
  - property names are in bold style – description
  - literals (values) are in green with a green background – complete
- All examples in this document are expressed in Consolas 9 pt font, with straight quotes and have a two space indentation. Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with the ellipses (...).
2. HTTPS Requirements

The TAXII Protocol defined in this specification requires HTTPS as the transport for all communications.

- TAXII Servers and Clients **MUST** implement HTTPS [RFC7230].
- TAXII Servers and Clients **MUST** implement TLS version 1.2 [RFC5246], and **MAY** implement later versions.
- The default strategy for authenticating certificates **MUST** be PKIX as defined in RFC 5280, RFC 6818, RFC 6125 et al.
- TAXII Servers and Clients **MAY** support other certification verification policies such as:
  - **Certificate Pinning**: A single or limited set of either hard-coded or physically distributed pinned certificate authorities or end-entity certificates.
  - **DANE**: DNS-based Authentication of Named Entities [RFC 7671]
  - Note that Self-Signed Certificates (like other certificates which cannot be verified by PKIX) **MAY** be supported via Certificate Pinning and/or DANE as noted above.

3. DNS SRV Records

This specification defines a DNS SRV record [RFC 2782] that can be used to allow clients to auto-discover the server that the TAXII server is running on.

3.1. Requirements

- Organizations **MAY** implement a DNS SRV record in their DNS server to advertise the location of their TAXII Server.
- The service name for this version of TAXII **MUST** be “taxii”.
  - Future versions of TAXII **MAY** define alternate service names.
- TAXII Clients **MUST** support looking up and using the TAXII SRV record from DNS.

3.2. Example

The following example is for a DNS SRV record advertising a TAXII server for the domain “example.com” located at taxii-hub-1.example.com:443:

```
_taxii._tcp.example.com. 86400 IN SRV 0 5 443 taxii-hub-1.example.com
```
4. Discovery API

This specification defines a Discovery API that clients can use to discover the capabilities that the TAXII Server offers as well as meta-information about the TAXII Server (e.g., contact information). This specification uses the notation `<discovery>` to refer to the Discovery API URL.

4.1. Requirements

- TAXII Servers **SHOULD** implement the Discovery API
- The URL of the Discovery API **MUST** be /taxii and **MUST** be located at the root of the server, e.g., https://someserver.com/taxii
- A TAXII Server **MAY** advertise TAXII services that are not running on this server.

4.2. Examples

https://taxii.example.com:443/taxii
https://someserver.foo.com/taxii

5. API Root

An API Root is the "root" URL of a particular instance of the TAXII API. Hosting multiple API Roots allows an implementer to mimic trust groups or groups of interest on a single TAXII Server.

5.1. Requirements

- A TAXII Server **MUST** host at least one API Root.
- A TAXII Server **MAY** host more than one API Root.
- Each API Root **MUST** have a unique URL within the scope of the TAXII Server.
- Each API Root **MAY** have different authentication requirements.

5.2. Examples

https://subdomain.example.com:12345/api-root-1
https://subdomain.example.com:12345/api-root-2
6. Content Negotiation

TAXII 2 uses HTTP content negotiation as defined below.

6.1. TAXII Media Type

This specification defines the media type for TAXII.

6.1.1. Requirements

- The TAXII media type representing any version of TAXII is: `application/vnd.oasis.taxii+json`
- The TAXII media type representing TAXII 2.0 is: `application/vnd.oasis.taxii+json; version=2.0`
- TAXII Clients **SHOULD** include the version token wherever a TAXII media type is used.
  - If the version token is omitted from a TAXII media type, implementations **SHOULD** respond with the highest version of TAXII that the server supports.
- TAXII Servers **MUST** honor the version token during content negotiation. If the server does not support the version, it should return a HTTP 406 (Not Acceptable) error.

6.2. STIX Media Type

This specification makes use of the STIX media type for representing Object and Message resources.

6.2.1. Requirements

- The STIX media type representing any version of STIX is: `application/vnd.oasis.stix+json`
- The STIX media type representing STIX 2.0 is: `application/vnd.oasis.stix+json; version=2.0`
- TAXII Clients **SHOULD** include the version token wherever a STIX media type is used.
  - If the version token is omitted from a STIX media type, implementations **SHOULD** respond with the highest version of STIX that the server supports.
- TAXII Servers **MUST** honor the version token during content negotiation. If the server does not support the version, it should return a HTTP 406 (Not Acceptable) error.
7. Primitive Types

This section defines the primitive types used throughout TAXII. These types will be referenced by the “Type” column in other sections. This section defines the names and permitted values of common types that are used in TAXII; it does not, however, define the meaning of any fields using these types. These types may be further restricted elsewhere in the document.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>A boolean is a value of either true or false. Properties with this type <strong>MUST</strong> have a literal (unquoted) value of true or false.</td>
</tr>
<tr>
<td>integer</td>
<td>The integer data type represents a whole number. Unless otherwise specified, all integers <strong>MUST</strong> be capable of being represented as a signed 64-bit value. Additional restrictions <strong>MAY</strong> be placed on the type as described where it is used.</td>
</tr>
<tr>
<td>list</td>
<td>The <strong>list</strong> type defines an ordered sequence of one or more values. The phrasing “list of type <code>&lt;type&gt;</code>” is used to indicate that all values within the list must conform to a specific type. For instance, <strong>list</strong> of type <strong>string</strong> means that all values of the list must be of the <strong>string</strong> type. This definition does not specify the maximum or minimum number of allowed values in a <strong>list</strong>, however specific TAXII resource properties may define more restrictive upper and/or lower bounds for the length of the list. If a <strong>list</strong> property is required but no data is available, then an empty <strong>list</strong> MUST be returned.</td>
</tr>
<tr>
<td>string</td>
<td>The <strong>string</strong> data type represents a finite-length string of valid characters from the Unicode coded character set [REF: ISO.10646] that are encoded in UTF-8. Unicode incorporates ASCII [REF: RFC20] and the characters of many other international character sets.</td>
</tr>
</tbody>
</table>
| timestamp | The **timestamp** type defines how timestamps are represented in TAXII and is represented in serialization as a **string**.  
  - The **timestamp** field **MUST** be a valid RFC 3339-formatted timestamp [TODO add reference] using the format YYYY-MM-DDTHH:mm:ss[.s+]Z where the “s+” represents 1 or more sub-second values. The brackets denote that sub-second values are optional. |
8. TAXII API

This section defines the TAXII API and all of the URL endpoints that are part of this specification.

8.1. URL Endpoint Summary

<table>
<thead>
<tr>
<th>Resource URL</th>
<th>Methods</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;discovery&gt;</td>
<td>GET</td>
<td>discovery</td>
</tr>
<tr>
<td>&lt;api-root&gt;</td>
<td>GET</td>
<td>api</td>
</tr>
<tr>
<td>Collections</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;api-root&gt;/status/&lt;status-id&gt;</td>
<td>GET</td>
<td>status</td>
</tr>
<tr>
<td>&lt;api-root&gt;/collections</td>
<td>GET</td>
<td>list of type collection</td>
</tr>
<tr>
<td>&lt;api-root&gt;/collections/&lt;name&gt;</td>
<td>GET</td>
<td>collection</td>
</tr>
<tr>
<td>&lt;api-root&gt;/collections/&lt;name&gt;/manifest</td>
<td>GET</td>
<td>list of type manifest</td>
</tr>
<tr>
<td>&lt;api-root&gt;/collections/&lt;name&gt;/objects</td>
<td>GET, POST</td>
<td>object*</td>
</tr>
<tr>
<td>&lt;api-root&gt;/collections/&lt;name&gt;/objects/&lt;object-id&gt;</td>
<td>GET</td>
<td>object*</td>
</tr>
<tr>
<td>&lt;api-root&gt;/object-search</td>
<td>GET</td>
<td>object*</td>
</tr>
<tr>
<td>&lt;TBD in a future RC&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The actual format of objects is dependent on HTTP Content negotiation, as discussed in Section [TODO REF]
8.2. URL Parameters

This section defines URL parameters and their meaning. The URL parameters defined in this section are used in the query portion of a URL. Each URL section defines which URL parameters are used.

<table>
<thead>
<tr>
<th>URL Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| **id**        | The identifier of the object that you are requesting. This is often a STIX ID. One or more identifiers **MAY** be specified in a single URL ID Parameter, comma separated.  
Example  
?id=1234,123,12334 |
| **type**      | The object type that you want to filter on. **TAXII Servers MUST** support the following values:  
- campaign  
- course-of-action  
- indicator  
- malware  
- relationship  
- report  
- sighting  
- threat-actor  
**TAXII Servers MAY** support other values.  
Implementers are strongly recommended to support all SDO and SRO types from all STIX versions that they support. |
| **version**   | The version of the STIX object that you are wanting.  
- **latest** tells the server to give you the latest one it knows about,  
- **first** tell the server to give you the first one it knows about.  
- **all** tells the server to give you all versions it knows about.  
- An actual version number aka "2" tells the server to give you version 2.  
- If the version parameter is not present in the request, it defaults to **latest**. |
8.3. Global Requirements

This section defines the behavior and requirements that apply globally to all URLs listed in this section.

8.3.1. Property and String Requirements

- All property names and string literals **MUST** be exactly the same, including case, as the names listed in the property tables in this specification.
  - For example, the discovery resource has a property called `api_roots` and it must result in the JSON key name "api_roots".
- Properties marked required in the property tables **MUST** be present in the JSON serialization.

8.3.2. Server Requirements

- TAXII Servers **MUST** implement all URLs and HTTP methods defined in this section.
  - TAXII Servers **MAY** implement other URLs and/or methods.
- TAXII Servers **MUST** include the version parameter in the Content-Type header when responding to Accept: headers of `application/vnd.oasis.taxii+json` and `application/vnd.oasis.stix+json;`.
  - For example: `Content-Type: application/vnd.oasis.taxii+json; version=2.0`
- The `status` resource **MUST** be returned when an HTTP 202 (Accepted) response is given to a POST request.
- A server generating an HTTP error response **SHOULD** also include the error message in the response payload to give additional application specific details about the error.
- Servers **MAY** silently ignore unauthorized requests from clients.
- If a TAXII Client is unauthorized to access one or more objects or resources in a returned list, the server **SHOULD** filter those records from the returned list instead of refusing to fulfill the request.

8.3.3. Client Requirements

- Requests **MUST** include an Accept: header.
- Requests that expect a STIX or TAXII response **SHOULD** include an appropriate media range in the accept header.
○ A media range of `application/vnd.oasis.taxii+json` in the accept header indicates that any version of TAXII is acceptable in the response.
○ A media range of `application/vnd.oasis.taxii+json; version=2.0` in the accept header indicates that ONLY TAXII 2.0 is acceptable in the response.
○ A media range of `application/vnd.oasis.stix+json` in the accept header indicates that any version of STIX is acceptable in the response.
○ A media range of `application/vnd.oasis.stix+json; version=2.0` in the accept header indicates that ONLY STIX 2.0 is acceptable in the response.

### 8.4. HTTP Status Codes

This section lists commonly used HTTP status codes as a reference for implementers. This specification does not modify the usage or meaning of HTTP status codes, and implementations are not restricted to using HTTP status codes listed in this section.

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Text Value</th>
<th>Notes (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP 200</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>HTTP 202</td>
<td>Accepted</td>
<td>The request was accepted but has not yet been processed. This is used when a group of Objects or Messages are POSTed, and the server will process them asynchronously.</td>
</tr>
<tr>
<td>HTTP 400</td>
<td>Bad Request</td>
<td></td>
</tr>
<tr>
<td>HTTP 401</td>
<td>Unauthorized</td>
<td></td>
</tr>
<tr>
<td>HTTP 403</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td>HTTP 404</td>
<td>Not Found</td>
<td></td>
</tr>
<tr>
<td>HTTP 405</td>
<td>Method Not Allowed</td>
<td>Each section defines requirements for certain HTTP Methods (e.g., GET, POST); each of these methods is said to be a supported method for the URL. HTTP requests that use a supported method MUST NOT result in an HTTP response with a status of 405 (Method Not Allowed). Other methods MAY result in an HTTP response with a status of 405. For example, the <code>&lt;api-root&gt;</code> defines requirements for the GET method, but not other methods. In this case, GET requests cannot result in an HTTP response with a status of 405; but POST requests may.</td>
</tr>
</tbody>
</table>
For HTTP responses that contain a message body, the format of the message body is negotiated using the HTTP Accept header. Formats that are specified in the HTTP Accept header that the server is capable of providing are said to be acceptable. Formats specified in the HTTP Accept header that the server is not capable of providing are said to be unacceptable.

If all options listed in the HTTP request’s Accept header are unacceptable, the HTTP response must have a status code of HTTP 406 (Not Acceptable). Each section defines which response formats must be acceptable.

For HTTP requests that contain a message body, the format of the message body is identified using the HTTP Content-Type header. For Content-Types that the server does not support, the HTTP response must have a status code of HTTP 415 (Unsupported Media Type). Each section defines which Content-Types must be supported. Additional Content-Types, beyond those listed, MAY be supported.

8.5. GET <discovery>

This URL allows TAXII Clients to discover information about a TAXII Server and learn the API Roots that this TAXII Server knows about.

The following tables illustrate a conformant request/response pair.

<table>
<thead>
<tr>
<th>HTTP 406</th>
<th>Not Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>For HTTP responses that contain a message body, the format of the message body is negotiated using the HTTP Accept header. Formats that are specified in the HTTP Accept header that the server is capable of providing are said to be acceptable. Formats specified in the HTTP Accept header that the server is not capable of providing are said to be unacceptable.</td>
<td></td>
</tr>
<tr>
<td>If all options listed in the HTTP request’s Accept header are unacceptable, the HTTP response must have a status code of HTTP 406 (Not Acceptable). Each section defines which response formats must be acceptable.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP 410</th>
<th>Gone</th>
</tr>
</thead>
<tbody>
<tr>
<td>For HTTP requests that contain a message body, the format of the message body is identified using the HTTP Content-Type header. For Content-Types that the server does not support, the HTTP response must have a status code of HTTP 415 (Unsupported Media Type). Each section defines which Content-Types must be supported. Additional Content-Types, beyond those listed, MAY be supported.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP 415</th>
<th>Unsupported Media Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>For HTTP responses that contain a message body, the format of the message body is negotiated using the HTTP Accept header. Formats that are specified in the HTTP Accept header that the server is capable of providing are said to be acceptable. Formats specified in the HTTP Accept header that the server is not capable of providing are said to be unacceptable.</td>
<td></td>
</tr>
<tr>
<td>If all options listed in the HTTP request’s Accept header are unacceptable, the HTTP response must have a status code of HTTP 406 (Not Acceptable). Each section defines which response formats must be acceptable.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP 429</th>
<th>Too Many Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>For HTTP requests that contain a message body, the format of the message body is identified using the HTTP Content-Type header. For Content-Types that the server does not support, the HTTP response must have a status code of HTTP 415 (Unsupported Media Type). Each section defines which Content-Types must be supported. Additional Content-Types, beyond those listed, MAY be supported.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP 500</th>
<th>Internal Server Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>For HTTP requests that contain a message body, the format of the message body is identified using the HTTP Content-Type header. For Content-Types that the server does not support, the HTTP response must have a status code of HTTP 415 (Unsupported Media Type). Each section defines which Content-Types must be supported. Additional Content-Types, beyond those listed, MAY be supported.</td>
<td></td>
</tr>
</tbody>
</table>

8.5. GET <discovery>

This URL allows TAXII Clients to discover information about a TAXII Server and learn the API Roots that this TAXII Server knows about.

The following tables illustrate a conformant request/response pair.

<table>
<thead>
<tr>
<th>Request Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
</tr>
</tbody>
</table>
8.5.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` **MUST** contain a JSON `discovery` object.

8.5.2. Examples

GET Request

```
GET /taxii HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

GET Response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0
{
  "display_name": "Some TAXII Server",
  "description": "This TAXII server contains a listing of...",
  "contact": "string containing contact information",
  "default": "https://example.com/api2",
}
8.6. GET <api-root>

This URL allows TAXII Clients to discover the available channels and collections at this specific API Root.

The following tables illustrate a conformant request/response pair.

<table>
<thead>
<tr>
<th>Request Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
<td>GET &lt;api-root&gt;</td>
</tr>
<tr>
<td>URL Variable(s)</td>
<td>&lt;api-root&gt; - the base URL of the API Root containing the collection</td>
</tr>
<tr>
<td>URL Parameters</td>
<td>n/a</td>
</tr>
<tr>
<td>Accept Header</td>
<td>application/vnd.oasis.taxii+json; version=2.0</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>n/a</td>
</tr>
<tr>
<td>Message Body</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Line</td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>application/vnd.oasis.taxii+json; version=2.0</td>
</tr>
<tr>
<td>Message Body</td>
<td>api-root</td>
</tr>
</tbody>
</table>

8.6.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
• Requests with an Accept header that contains `application/vnd.oasis.taxii+json` MUST NOT result in an HTTP 406 (Not Acceptable) response.

• HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` MUST contain a JSON `api-root` object.

• Servers MAY elide results from the response. For example, a server might choose to elide an API root if the requestor did not have sufficient permissions to view it.

8.6.2. Examples

GET Request

```
GET /some-api-base HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

GET Response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0
{
  "display_name": "Malware Research Group",
  "description": "A trust group setup for malware researchers",
  "versions": ["taxii-2.0"],
  "channels": [],
  "collections": [
    {
      "url": "https://example.com/api-1/collections/high-value-indicators",
      "display_name": "High Value Indicator Collection",
      "description": "This data collection is for collecting high value IOCs",
      "can_read": true,
      "can_write": false,
      "media_types": [
        "application/vnd.oasis.stix+json; version=2.0"
      ],
      "objects_count": 923
    },
    {
      "url": "https://example.com/tg1/collections/24-hour-indicators",
      "display_name": "Indicators from the past 24-hours",
      "description": "This data collection is for collecting current IOCs",
      "can_read": true,
      "can_write": false,
      "media_types": [
        "application/vnd.oasis.stix+json; version=2.0"
      ],
      "objects_count": 7
    }
  ],
  "max_content_length": 9765625,
}
```
8.7. GET /collections

This URL allows TAXII Clients to get a list of Collection resources that are available within an API Root.

The following tables illustrate a conformant request/response pair.

### Request Properties

<table>
<thead>
<tr>
<th>Request Line</th>
<th>GET &lt;api-root&gt;/collections</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Variable(s)</td>
<td>&lt;api-root&gt; - the base URL of the API Root containing the collection</td>
</tr>
<tr>
<td>URL Parameters</td>
<td>n/a</td>
</tr>
<tr>
<td>Accept Header</td>
<td>application/vnd.oasis.taxii+json; version=2.0</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>n/a</td>
</tr>
<tr>
<td>Message Body</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Response Properties

<table>
<thead>
<tr>
<th>Status Line</th>
<th>HTTP/1.1 200 OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type Header</td>
<td>application/vnd.oasis.taxii+json; version=2.0</td>
</tr>
<tr>
<td>Message Body</td>
<td>list of type collection</td>
</tr>
</tbody>
</table>

### 8.7.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` **MUST** contain a JSON list, where each item in the list is a `collection`.
  - If there are zero `collection` to return, the result is an empty list.
  - If there is one `collection` to return, the result is a list with one item.
Servers MAY elide results from the response. For example, a server might choose to elide a collection if the requestor did not have sufficient permissions to view it.

8.7.2. Examples

GET Request

```
GET https://example.com/api-1/collections HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

GET Response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0

[
   {
       "url": "https://example.com/api-1/collections/high-value-indicators",
       "display_name": "High Value Indicator Collection",
       "description": "This data collection is for collecting high value IOCs",
       "can_read": true,
       "can_write": false,
       "media_types": [
           "application/vnd.oasis.stix+json; version=2.0"
       ],
       "objects_count": 923
   },
   {
       "url": "https://example.com/tg1/collections/24-hour-indicators",
       "display_name": "Indicators from the past 24-hours",
       "description": "This data collection is for collecting current IOCs",
       "can_read": true,
       "can_write": false,
       "media_types": [
           "application/vnd.oasis.stix+json; version=2.0"
       ],
       "objects_count": 7
   }
]
```

8.8. GET /collections/<name>

This URL allows TAXII Clients to get details about this specific Collection.

The following tables illustrate a conformant request/response pair.

```
<table>
<thead>
<tr>
<th>Request Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
</tr>
<tr>
<td>GET &lt;api-root&gt;/collections/&lt;name&gt;</td>
</tr>
</tbody>
</table>
```

Figure XX - Collection Request Properties
### URL Variable(s)
- `<api-root>` - the base URL of the API Root containing the collection
- `<name>` - the name of the collection being requested

### URL Parameters
n/a

### Accept Header
`application/vnd.oasis.taxii+json; version=2.0`

### Content-Type Header
n/a

### Message Body
n/a

---

**Figure XX - Collection Response Properties**

<table>
<thead>
<tr>
<th>Response Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Line</strong></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td><strong>Content-Type Header</strong></td>
<td><code>application/vnd.oasis.taxii+json; version=2.0</code></td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
<td><code>collection</code></td>
</tr>
</tbody>
</table>

---

**Figure XX - Common Errors**

<table>
<thead>
<tr>
<th>Common Errors</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Code</strong></td>
<td>Possible Reason</td>
</tr>
<tr>
<td>HTTP 404</td>
<td>The Collection is not found</td>
</tr>
</tbody>
</table>

---

### 8.8.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` **MUST** contain a JSON `collection` object.
  - If no `collection` is returned, the result is an HTTP 404 (Not Found).
- Servers **MAY** elide results from the response. For example, a server might choose to elide a collection if the requestor did not have sufficient permissions to view it.
8.8.2. Examples

GET Request

```
GET https://example.com/api-1/collections/high-value-indicators HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

GET Response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0
{
  "url": "https://example.com/api-1/collections/high-value-indicators",
  "display_name": "High Value Indicator Collection",
  "description": "This data collection is for collecting high value IOCs",
  "can_read": true,
  "can_write": false,
  "media_types": [
    "application/vnd.oasis.stix+json; version=2.0"
  ],
  "objects_count": 923
}
```

8.9. GET /collections/<name>/manifest

This URL allows TAXII Clients to get a Manifest of Objects in this collection. This URL supports URL parameters to filter the results.

The following tables illustrate a conformant request/response pair.

| Request Properties |  
|--------------------|---|
| **Request Line**   | GET <api-root>/collections/<name>/manifest |
| **URL Variable(s)**| `<api-root>` - the base URL of the API Root containing the collection `<name>` - the name of the collection being requested |
| **URL Parameters** | `<id` `<type` `<version` `<added_after` |
| **Accept Header**  | application/vnd.oasis.taxii+json; version=2.0 |
| **Content-Type**   | n/a |
8.9.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` **MUST** contain a JSON list, where each item in the list is a `manifest`.
  - If there are zero `manifest` to return, the result is an empty list.
  - If there is one `manifest` to return, the result is a list with one item.
- Servers **MAY** elide results from the response. For example, a server might choose to elide a manifest if the requestor did not have sufficient permissions to view it.
- The `id`, `type`, `version`, and `added_after` URL parameters **MUST** be supported at this URL endpoint.

8.9.2. Examples

**GET Request**

```
GET https://example.com/api-1/collections/high-value-indicators/manifest HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

**GET Response**

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0

[  
  {
      "url": "https://example.com/api-1/collections/high-value-indicators/objects/indicator--c410e480-...-85307c12bcbf",
      "date_added": "2016-11-01T03:04:05Z",
  }
]```
8.10. GET /collections/<name>/objects

This URL allows TAXII Clients to get multiple Objects in this collection. This URL supports URL parameters to filter the results.

The following tables illustrate a conformant request/response pair.

**Request Properties**

<table>
<thead>
<tr>
<th>Request Line</th>
<th>GET &lt;api-root&gt;/collections/&lt;name&gt;/objects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URL Variable(s)</strong></td>
<td></td>
</tr>
<tr>
<td>&lt;api-root&gt;</td>
<td>- the base URL of the API Root containing the collection</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>- the name of the collection being requested</td>
</tr>
<tr>
<td><strong>URL Parameters</strong></td>
<td>id, type, version, added_after</td>
</tr>
<tr>
<td><strong>Accept Header</strong></td>
<td>application/vnd.oasis.stix+json; version=2.0</td>
</tr>
<tr>
<td><strong>Content-Type</strong></td>
<td>n/a</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>n/a</td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Response Properties**

<table>
<thead>
<tr>
<th>Status Line</th>
<th>HTTP/1.1 200 OK</th>
</tr>
</thead>
</table>
8.10.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.stix+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.stix+json` **MUST** contain an `object` resource.
  - If there are no `object` to return, the result is an HTTP 404 (Not Found).
- Servers **MAY** elide results from the response. For example, a server might choose to elide a manifest if the requestor did not have sufficient permissions to view it.
- The `id`, `type`, `version`, and `added_after` URL parameters **MUST** be supported at this URL endpoint.

8.10.2. Examples

**GET Request**

```
GET https://example.com/api-1/collections/high-value-indicators/objects HTTP/1.1
Accept: application/vnd.oasis.stix+json
```

**GET Response**

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.stix+json; version=2.0
{
  "type": "bundle",
  ...,
  "indicators": [
    {
      "type": "indicator",
      ...,
    }
  ]
}
```
8.11. POST /collections/<name>/objects

This URL allows TAXII Clients to create an Object in this collection.

The following tables illustrate a conformant request/response pair.

**Figure XX - Collection Objects Request Properties**

<table>
<thead>
<tr>
<th>Request Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request Line</strong></td>
</tr>
<tr>
<td><strong>URL Variable(s)</strong></td>
</tr>
<tr>
<td><strong>URL Parameters</strong></td>
</tr>
<tr>
<td><strong>Accept Header</strong></td>
</tr>
<tr>
<td><strong>Content-Type Header</strong></td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
</tr>
</tbody>
</table>

**Figure XX - Collection Objects Response Properties**

<table>
<thead>
<tr>
<th>Response Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Line</strong></td>
</tr>
<tr>
<td><strong>Content-Type Header</strong></td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
</tr>
</tbody>
</table>

8.11.1. Requirements

- A TAXII Server **MUST** support POST requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- Requests with a Content-Type header that contains `application/vnd.oasis.stix+json` **MUST NOT** result in an HTTP 415 (Unsupported Media Type) response.
HTTP 202 Responses with a Content-Type of application/vnd.oasis.taxii+json MUST contain an status resource.

- The client should periodically poll the URL contained in the status url property to retrieve the most up-to-date status, until such a time that the status property returns a value of complete.
- TAXII Servers SHOULD NOT delete status messages for at least 24 hours.

8.11.2. Examples

POST Request

```
POST https://example.com/api-1/collections/high-value-indicators/objects HTTP/1.1
Accept: application/vnd.oasis.taxii+json
Content-Type: application/vnd.oasis.stix+json; version=2.0
{
  "type": "bundle",
  ...
  "indicators": [
    {"type": "indicator",
     "id": "indicator--c410e480-e42b-47d1-9476-85307c12bcbf",
     ...
    }
  ]
}
```

POST Response

```
HTTP/1.1 202 Accepted
Content-Type: application/vnd.oasis.taxii+json; version=2.0
{
  "url": "https://example.com/api-1/status/1234",
  "status": "pending",
  "request_url": "https://example.com/api-1/collections/coll1/objects",
  "request_timestamp": "2016-11-02T12:34:34.12345Z",
  "total_items": 1,
  "success_count": 1,
  "success_items": [
    {"id": "indicator--c410e480-e42b-47d1-9476-85307c12bcbf",
     "url": "https://example.com/api-1/collections/coll1/objects/indicator--c410e480-e42b-47d1-9476-85307c12bcbf"
    }
  ]
}
```

8.12. GET /collections/<name>/objects/<object-id>

This URL allows TAXII Clients to get a specific Object from this collection. This URL only supports the version URL parameters to filter the results.
The following tables illustrate a conformant request/response pair.

**Figure XX - Collection Objects ID Request Properties**

<table>
<thead>
<tr>
<th>Request Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
<td>GET &lt;api-root&gt;/collections/&lt;name&gt;/objects/&lt;object-id&gt;</td>
</tr>
<tr>
<td>URL Variable(s)</td>
<td>&lt;api-root&gt; - the base URL of the API Root containing the collection&lt;br&gt; &lt;name&gt; - the name of the collection being requested&lt;br&gt; &lt;object-id&gt; - the ID of the object being requested</td>
</tr>
<tr>
<td>URL Parameters</td>
<td>version</td>
</tr>
<tr>
<td>Accept Header</td>
<td>application/vnd.oasis.stix+json; version=2.0</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>n/a</td>
</tr>
<tr>
<td>Message Body</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Figure XX - Collection Objects ID Response Properties**

<table>
<thead>
<tr>
<th>Response Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Line</td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>application/vnd.oasis.stix+json; version=2.0</td>
</tr>
<tr>
<td>Message Body</td>
<td>object*</td>
</tr>
</tbody>
</table>

*The actual format of objects is dependent on HTTP Content negotiation, as discussed in Section [TODO REF]*

8.12.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.stix+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.stix+json` **MUST** contain an `object` resource.
  - If no `object` is returned, the result is an HTTP 404 (Not Found).
- Servers **MAY** elide results from the response. For example, a server might choose to elide an object if the requestor did not have sufficient permissions to view it.
- The `version` URL parameter **MUST** be supported at this URL endpoint.
8.12.2. Examples

GET Request

```plaintext
GET https://example.com/api-1/collections/high-value-indicators/object/indicator--252c7c11-daf2-42bd-843b-be65edca9f61 HTTP/1.1
Accept: application/vnd.oasis.stix+json; version=2.0
```

GET Response

```plaintext
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.stix+json; version=2.0
{
    "type": "bundle",
    "indicators": [
        {
            "type": "indicator",
            "id": "indicator--252c7c11-daf2-42bd-843b-be65edca9f61",
            ...
        }
    ]
}
```

8.13. GET /object-search

This URL allows TAXII Clients to retrieve multiple Objects from any collection in this API Root. This URL supports URL parameters to filter the results.

The following tables illustrate a conformant request/response pair.

```
<table>
<thead>
<tr>
<th>Request Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
<td>GET &lt;api-root&gt;/object-search</td>
</tr>
<tr>
<td>URL Variable(s)</td>
<td>&lt;api-root&gt; - the base URL of the API Root</td>
</tr>
<tr>
<td>URL Parameters</td>
<td>id, type, version, added_after</td>
</tr>
<tr>
<td>Accept Header</td>
<td>application/vnd.oasis.stix+json; version=2.0</td>
</tr>
<tr>
<td>Content-Type Header</td>
<td>n/a</td>
</tr>
</tbody>
</table>
```

Figure XX - Object Search Request Properties
**Figure XX - Object Search Response Properties**

<table>
<thead>
<tr>
<th>Response Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Line</strong></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td><strong>Content-Type Header</strong></td>
<td>application/vnd.oasis.stix+json; version=2.0</td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
<td>object*</td>
</tr>
</tbody>
</table>

* The actual format of objects is dependent on HTTP Content negotiation, as discussed in Section [TODO REF]

### 8.13.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.stix+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.stix+json` **MUST** contain an `object` resource.
  - If no `object` is returned, the result is an HTTP 404 (Not Found).
- Servers **MAY** elide results from the response. For example, a server might choose to elide an object if the requestor did not have sufficient permissions to view it.
- The `id`, `type`, `version`, and `added_after` URL parameters **MUST** be supported at this URL endpoint.

### 8.13.2. Examples

**GET Request**

```
GET https://example.com/api-1/object-search?type=indicator HTTP/1.1
Accept: application/vnd.oasis.stix+json; version=2.0
```

**GET Response**

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.stix+json; version=2.0
{
   "type": "bundle",
   
   "indicators": [
      {
         "type": "indicator",
         
      }
   ]
}
8.14. GET /status/<status-id>

This URL allows TAXII Clients to get a Status of a previous request. This is used to monitor the status of requests that have resulted in an HTTP 202 (Accepted) response.

The following tables illustrate a conformant request/response pair.

---

**Figure XX - Status Request Properties**

<table>
<thead>
<tr>
<th>Request Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request Line</strong></td>
<td>GET &lt;api-root&gt;/status/&lt;status-id&gt;</td>
</tr>
</tbody>
</table>
| **URL Variable(s)**| <api-root> - the base URL of the API Root  
                  | <status-id> - the ID of the status message being requested |
| **URL Parameters** | n/a |
| **Accept Header**  | application/vnd.oasis.taxii+json; version=2.0 |
| **Content-Type Header** | n/a |
| **Message Body**   | n/a |

---

**Figure XX - Status Response Properties**

<table>
<thead>
<tr>
<th>Response Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Line</strong></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td><strong>Content-Type Header</strong></td>
<td>application/vnd.oasis.taxii+json; version=2.0</td>
</tr>
<tr>
<td><strong>Message Body</strong></td>
<td>status</td>
</tr>
</tbody>
</table>

---

**Figure XX - Common Errors**

<table>
<thead>
<tr>
<th>Common Errors</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status Code</strong></td>
<td><strong>Possible Reason</strong></td>
</tr>
<tr>
<td>HTTP 404</td>
<td>The Status ID is not found</td>
</tr>
</tbody>
</table>
8.14.1. Requirements

- A TAXII Server **MUST** support GET requests at this URL and **MAY** support other request types.
- Requests with an Accept header that contains `application/vnd.oasis.taxii+json` **MUST NOT** result in an HTTP 406 (Not Acceptable) response.
- HTTP 200 Responses with a Content-Type of `application/vnd.oasis.taxii+json` **MUST** contain a `status` resource.
  - If no `status` is returned, the result is an HTTP 404 (Not Found).
- Servers **MAY** elide results from the response. For example, a server might choose to elide the status if the requestor did not have sufficient permissions to view it.

8.14.2. Examples

**GET Request**

```
GET /some-api-root/status/123456 HTTP/1.1
Accept: application/vnd.oasis.taxii+json
```

**GET Response**

```
HTTP/1.1 200 OK
Content-Type: application/vnd.oasis.taxii+json; version=2.0
{
    "url": "https://example.com/api-1/status/1234",
    "status": "pending",
    "request_url": "https://example.com/api-1/collections/coll1/objects",
    "request_timestamp": "2016-11-02T12:34:34.12345Z",
    "total_items": 4,
    "success_count": 1,
    "success_items": [
        { "id": "indicator--c410e480-e42b-47d1-9476-85307c12bcbf",
          "url": "https://example.com/api-1/collections/coll1/objects/indicator--c410e480-e42b-47d1-9476-85307c12bcbf"
        }
    ],
    "failure_count": 1,
    "failure_items": [
        { "id": "malware--664fa29d-bf65-4f28-a667-bdb76f29ec98",
          "message": "Unable to process object"
        }
    ],
    "pending_count": 2,
    "pending_items": [
        "indicator--252c7c11-daf2-42bd-843b-bee65edca9f61",
        "relationship--045585ad-a22f-4333-af33-bfd583ab83b5"
    ]
}
```
9. TAXII Resources

This section defines the TAXII resources and their representations.

9.1. API Root Resource

**Resource Name:** api-root

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display_name (required)</td>
<td>string</td>
<td>A human readable text/plain name used to identify this API instance. This is not the name of this API Root that is found in the URL.</td>
</tr>
<tr>
<td>description (optional)</td>
<td>string</td>
<td>A human readable text/plain description for this API Root.</td>
</tr>
<tr>
<td>versions (required)</td>
<td>list of type</td>
<td>Lists the versions of TAXII that this API Root is compatible with. taxii-2.0 MUST be included in this list to indicate conformance with this specification.</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>channels (required)</td>
<td>list of type</td>
<td>&lt;TODO&gt;</td>
</tr>
<tr>
<td></td>
<td>channel</td>
<td></td>
</tr>
<tr>
<td>collections (required)</td>
<td>list of type</td>
<td>&lt;TODO&gt;</td>
</tr>
<tr>
<td></td>
<td>collection</td>
<td></td>
</tr>
<tr>
<td>max_content_length (optional)</td>
<td>integer</td>
<td>The maximum value of the request body &quot;Content-Length&quot; in octets (8-bit bytes) that the server can support. This applies to requests and responses and is determined by the server. Requests or responses with total body lengths values smaller than this value MUST NOT result in an HTTP 413 (Request Entity Too Large) response. Absence of this value means the server is choosing to not provide this information. This property is needed to help make sure a client does not have to keep guessing at how much data it can send. For example, if a server only supports payloads up to 10MB and a client wants to send 50MB of data, without this, the client would have to keep guessing as to what the server will support.</td>
</tr>
</tbody>
</table>
### 9.2. Discovery Resource

**Resource Name:** discovery

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display_name</td>
<td>string</td>
<td>A human readable text/plain name used to identify this server.</td>
</tr>
<tr>
<td>description (optional)</td>
<td>string</td>
<td>A human readable text/plain description for this server.</td>
</tr>
<tr>
<td>contact (optional)</td>
<td>string</td>
<td>The human readable text/plain contact information for this server and or the administrator of this server.</td>
</tr>
<tr>
<td>default (optional)</td>
<td>string</td>
<td>The default API Root that a TAXII Client <strong>MAY</strong> use. Absence of this field indicates that there is no default API Root.</td>
</tr>
<tr>
<td>api_roots (required)</td>
<td>list of type string</td>
<td>A list of URLs that identify API Roots that are hosted on this server or that this server knows about. This list <strong>MAY</strong> be filtered on a per-client basis.</td>
</tr>
</tbody>
</table>

### 9.3. Collection Resource

**Resource Name:** collection

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url (optional)</td>
<td>string</td>
<td>The full URL of this collection. This property <strong>MUST</strong> be present in GET responses.</td>
</tr>
<tr>
<td>display_name</td>
<td>string</td>
<td>A human readable text/plain name used to identify this Collection.</td>
</tr>
<tr>
<td>description (optional)</td>
<td>string</td>
<td>A human readable text/plain description for this Collection.</td>
</tr>
<tr>
<td>can_read (optional)</td>
<td>boolean</td>
<td>Indicates if the requester can read (i.e., GET) objects from this Collection. Absence of this field is equivalent to a value of <strong>false</strong>.</td>
</tr>
<tr>
<td>can_write (optional)</td>
<td>boolean</td>
<td>Indicates if the the requester can write (i.e., POST) objects to this Collection. Absence of this</td>
</tr>
</tbody>
</table>
9.4. Error Resource

Resource Name: error

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_id (optional)</td>
<td>string</td>
<td>An identifier for this particular error. A TAXII Server might choose to assign each error occurrence its own identifier in order to facilitate debugging.</td>
</tr>
<tr>
<td>display_name (required)</td>
<td>string</td>
<td>A human readable text/plain name describing this error.</td>
</tr>
<tr>
<td>description (optional)</td>
<td>string</td>
<td>A human readable text/plain description that gives details about the error or problem that was encountered by the application.</td>
</tr>
<tr>
<td>error_code (optional)</td>
<td>string</td>
<td>The error code for this error. A TAXII Server might choose to assign a common error code to all errors of the same type.</td>
</tr>
<tr>
<td>http_status (optional)</td>
<td>string</td>
<td>The HTTP status code applicable to this error.</td>
</tr>
<tr>
<td>external_details (optional)</td>
<td>string</td>
<td>A URL that points to additional details. Absence of this field indicates that there are not additional details.</td>
</tr>
<tr>
<td>details (optional)</td>
<td>object</td>
<td>The details objects provide a location for additional application specific details.</td>
</tr>
</tbody>
</table>

9.4.1. Example

```json
{
    "error_id": "1234",
    "display_name": "Error condition XYZ",
    "description": "This error is caused when the application tries to access data...",
    "error_code": "581234",
```
9.5. Manifest Resource

**Resource Name:** manifest

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url (required)</td>
<td>string</td>
<td>The full URL of this object</td>
</tr>
<tr>
<td>date_added (optional)</td>
<td>timestamp</td>
<td>The date this object was added to the server.</td>
</tr>
<tr>
<td>last_modified (optional)</td>
<td>timestamp</td>
<td>The date this object was last updated or the last modified date of the most current version. For example, if version 2 was added after version 3, this date would contain the last modified date from version 3 not version 2 as version 3 is the most current version.</td>
</tr>
<tr>
<td>versions (optional)</td>
<td>list of type string</td>
<td>A list of available STIX versions. This field is only meaningful for objects that are available in a STIX format.</td>
</tr>
<tr>
<td>media_types (optional)</td>
<td>list of type string</td>
<td>The media types that this object can be requested in.</td>
</tr>
</tbody>
</table>

9.6. Object Resource

**Resource Name:** object

This resource type is negotiated by the media type. If the media type in the Accept or Content-Type header contains application/vnd.oasis.stix+json; version=2.0 then this resource type is a STIX Bundle version 2.0 as defined in the STIX specification located here [TODO add reference].

9.7. Status Resource

**Resource Name:** status
This resource is returned when an HTTP 202 (Accepted) response is given to a POST request. This resource conveys the status of a bulk creation.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url (required)</td>
<td>string</td>
<td>The full URL of the this Status resource. Absence of this field means that there is no URL to get this status again.</td>
</tr>
<tr>
<td>status (required)</td>
<td>string</td>
<td>The overall status of a previous POST request where an HTTP 202 (Accept) was returned. The value of this property MUST be one of complete or pending. A value of complete indicates that this resource will not be updated further and MAY be removed in the future. A status of pending indicates that this resource MAY update in the future.</td>
</tr>
<tr>
<td>request_url (optional)</td>
<td>string</td>
<td>The URL of the request that this status resource is monitoring.</td>
</tr>
<tr>
<td>request_timestamp (optional)</td>
<td>timestamp</td>
<td>The datetime of the request that this status resource is monitoring.</td>
</tr>
<tr>
<td>total_items (required)</td>
<td>integer</td>
<td>The total number of items that were in the request. For a STIX Bundle this would be the number of objects in the Bundle.</td>
</tr>
<tr>
<td>success_count (optional)</td>
<td>integer</td>
<td>The number of items that were successfully created. Absence of this field is equivalent to a value of “0” (zero).</td>
</tr>
<tr>
<td>success_items (optional)</td>
<td>list of type success-item</td>
<td>A list of items that were successfully processed.</td>
</tr>
<tr>
<td>failure_count (optional)</td>
<td>integer</td>
<td>The number of items that failed to be created. Absence of this field is equivalent to a value of “0” (zero).</td>
</tr>
<tr>
<td>failure_items (optional)</td>
<td>list of type failure-item</td>
<td>A list of items that were not successfully processed.</td>
</tr>
<tr>
<td>pending_count (optional)</td>
<td>integer</td>
<td>The number of items that have yet to be processed.</td>
</tr>
</tbody>
</table>
### pending_items (optional)

| list of type string | A list of identifiers for items that have yet to be processed. |

**Type Name: success-items**

This type contains a list of success items by ID and location.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (required)</td>
<td>string</td>
<td>The identifier of the item that was created. For types that have an identifier, that identifier should be used here.</td>
</tr>
<tr>
<td>url (required)</td>
<td>string</td>
<td>The URL location of the created item.</td>
</tr>
</tbody>
</table>

**Type Name: failure-items**

This type contains a list of success items by ID and location.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (required)</td>
<td>string</td>
<td>The identifier of the item that was created. For types that have an identifier, that identifier should be used here.</td>
</tr>
<tr>
<td>message (optional)</td>
<td>string</td>
<td>A message indicating why the item failed to be created.</td>
</tr>
</tbody>
</table>

## 10. Customizing TAXII Resources

### 10.1. Custom Properties

It is understood that there will be cases where certain information exchanges can be improved by adding properties that are not specified nor reserved in this document; these properties are called **Custom Properties**. This section provides guidance and requirements for how TAXII Servers can use Custom Properties and how TAXII Clients should interpret them in order to extend TAXII in an interoperable manner.
10.1.1. Requirements

- A TAXII resource **MAY** have any number of Custom Properties.
- Custom Property names **MUST** be in ASCII and are limited to characters a-z (lowercase ASCII) and underscore (_).
- Custom Property names **SHOULD** start with “x_” followed by a source unique identifier (like a domain name), an underscore and then the name. For example: `x_examplecom_customfield`.
- Custom Property names **SHOULD** be no longer than 30 ASCII characters in length.
- Custom Property names **MUST** have a minimum length of 3 ASCII characters.
- Custom Property names **MUST** be no longer than 256 ASCII characters in length.
- Custom Property names that are not prefixed with “x_” may be used in a future version of the specification for a different meaning. If compatibility with future versions of this specification is required, the “x_” prefix **MUST** be used.
- Custom Property names **SHOULD** be unique when produced by the same source and **SHOULD** use a consistent namespace prefix (e.g., a domain name).
- Custom Properties **SHOULD** only be used when there is no existing properties defined by the TAXII specification that fulfills that need.

A TAXII Client that receives a TAXII message with one or more Custom Properties it does not understand **MAY** refuse to process the message further, or silently ignore non-understood properties and continue processing the message.

The reporting and logging of errors originating from the processing of Custom Properties depends heavily on the TAXII Server and Client implementations and is therefore not covered in this specification.

*Non-Normative*: TAXII Servers that produce messages that contain Custom Properties should be aware of the variability of TAXII Client behavior depending on whether or not the TAXII Client understands the Custom Properties present in a TAXII message. Rules for processing Custom Properties should be well defined and accessible to any TAXII Client that would be reasonably expected to parse them.

**Examples**

```json
{
  ...
  "x_acmeinc_scoring": {
    "impact": "high",
    "probability": "low"
  },
  ...
}
```
11. Conformance

11.1. TAXII Servers

A "TAXII 2.0 Server" is any software that creates publishes CTI content and conforms to the following normative requirements:

1. It **MUST** communicate over HTTPS using at least TLS version 1.2.
2. It **MUST** be able to create content encoded as JSON.
3. All required properties **MUST** be present in the created content.
4. All properties **MUST** conform to the data type and normative requirements for that property.
5. It **MUST** support at least Collections or Channels.
6. It **MUST** support all features listed in Section 11.2, Mandatory Features.
7. It **MAY** support any features listed in Section 11.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.

A "TAXII 2.0 Client" is any software that consumes CTI content and conforms to the following normative requirements:

1. It **MUST** communicate over HTTPS using at least TLS version 1.2.
2. It **MUST** support parsing all required properties for the content that it consumes.
3. It **MUST** support all features listed in Section 8.2, Mandatory Features.
4. It **MAY** support any features listed in Section 8.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.

11.2. Mandatory Features

11.2.1. TODO

A TAXII 2.0 Server or TAXII 2.0 Client **MUST** support X by following the normative requirements listed in Section X.

11.3. Optional Features

11.3.1. TODO

A TAXII 2.0 Server or TAXII 2.0 Client **MAY** support X. Software claiming to support X **MUST** follow the normative requirements listed in Section X and X.

11.3.2. Granular Data Markings
12. Appendix A. Acknowledgments

TAXII Subcommittee Chairs:
Bret Jordan (bret.jordan@bluecoat.com), Blue Coat Systems, Inc.
Mark Davidson (mdavidson@soltra.com), Soltra

Special Thanks:
The following individuals made substantial contributions to this specification in the form of normative text and proofing and their contributions are gratefully acknowledged:

- Bret Jordan, Blue Coat Systems, Inc.
- Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)
- Eric Burger, Georgetown University
- Jason Keirstead, IBM
- Mark Davidson, Soltra
- Allan Thomson, LookingGlass Cyber
- John Wunder, MITRE Corporation
- John-Mark Gurney, New Context Services, Inc.

Contributors:
The following individuals were members of the OASIS CTI Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:
<todo, make sure this is up to date>

13. Appendix B. Changes from TAXII 1.1

For the TAXII 2.x series of releases, TAXII is no longer an acronym; TAXII is now just “TAXII”. TAXII previously expanded to “The Trusted Automated eXchange of Indicator Information”. TAXII’s use has proven to be broader than indicators, and therefore the original name became artificially limiting.

Network level discovery is added through the definition of a DNS SRV record. Prior versions of TAXII did not have network level discovery.

HTTP Long Polling is explicitly defined in an attempt to reduce time-delay in server-push scenarios (with the understanding that HTTP is not well-suited to server pushes of information). TAXII 1.x was silent on HTTP Timeouts.

Authentication requirements are specified in an attempt to improve interoperability. Previous versions of TAXII were silent on authentication requirements.
The protocol and format are explicitly defined in one document (this document). Previous versions of TAXII used one document per protocol/format definition, resulting in multiple documents per TAXII release. This change is the result of removing a requirement that TAXII be implementable across any protocol and format combination.

TAXII is now explicitly JSON over HTTPS. Previous versions of TAXII were flexible regarding protocol and format.

While TAXII 2 maintains all the capabilities of previous releases, many capabilities have been transformed into their RESTful equivalents. Those changes are:

- TAXII Services have been removed as a concept. In their place are RESTful interactions (e.g., HTTP GET, HTTP POST).
- TAXII Data Sets have been formalized as containers of content. This specification calls them TAXII Collections.
- TAXII Data Feeds have been formalized as a Publish/Subscribe messaging pattern. This specification calls them TAXII Channels.

14. Appendix C - End to End Workflow

When this Discovery API is used with a DNS SRV record, clients auto-discover TAXII services as follows:

**Step 1:** Client uses DNS to retrieve a TAXII DNS Service Record, example:

```
_taxii._tcp.example.com. 86400 IN SRV 0 5 443 taxii.example.com
```

If no SRV record is present for the domain, the client assumes the TAXII service is available at the domain itself, and the default HTTPS port of 443.

**Step 2:** Client uses the TAXII information provided in the DNS Service Record to construct the Discovery API URL. Using the port (443) and the hostname (taxii.example.com) from the DNS SRV Record, the TAXII Discovery URL is constructed as follows:

```
https://hostname:port/taxii/
```

**Step 3:** Client issues a request to the fully constructed TAXII Discovery URL:

```
https://taxii.example.com:443/taxii/
```
1. Discuss DNS discovery
2. Get Discovery
3. Follow default to API Root
4. List collections
5. Pick collection to interact with
6. List objects
7. Get an object
8. Create an object

15. Appendix D - Security Considerations

This appendix should talk about security issues with DNS SRV Records. For TAXII Server the reference identifier expected in certificates SHALL be the original name, as per RFC 6125. The concern has to do with a DNS query to example.com and then getting back a result of taxii-hub-1.example.com and the client then making a followon request. Dave Cridland can give more details here. The problem we are trying to address is that if you do a DNS lookup for the domain example.com and it returns a SRV record for taxii.example.com, there is no way to verify that record without DNSSEC. 1) remove 2) just call out that you SHOULD use DNSSEC 3) talk about the end server needing a cert for all of the names aka example.com + foo.example.com