TAXII 2.0 Specification Pre-Draft

Current Status/Intent

This document serves to gain consensus on pre-draft concepts of TAXII 2.0. Please feel free to poke holes and comment!

Overview

TAXII is an open 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.

Changes from TAXII 1.1

TAXII is no longer an acronym. TAXII is just TAXII.

Table of Contents

- Current Status/Intent
- Overview
- Changes from TAXII 1.1
- Table of Contents
- TAXII Server
  - Broker Concepts
  - Repository Concepts
- TAXII API
  - API Base
  - URL Reference
  - HTTP Status Code Reference
  - HTTP X-Headers
    - URL Requirements
    - [API-Base]
  - Repository
- DNS SRV
  - Default API Base:
  - Discovery URL:
- Messages

- API-Base Information
- Channel Listing
- TAXII Messages (?)
- Repository Listing

Messaging Considerations

BELOW HERE IS NOT PART OF THE PRE-DRAFT AND IS JUST TEXT THAT IS KEPT FOR FUTURE REFERENCE AND WILL BE DELETED LATER

TAXII Broker

TAXII Server

- HTTPS
- Authentication

TAXII Repository Service

RESTful API

- TAXII Server API
  - API-Base
- TAXII Broker API
  - Broker API Methods
    - GET Channels
    - Get Channel <name>
- TAXII Repository API

TAXII Server

This section describes what a TAXII Server is and how it behaves. A TAXII Server is a piece of cyber security infrastructure that facilitates the exchange of cyber threat information across product and organizational boundaries. TAXII Servers facilitate the exchange of cyber threat information in two ways: as a message broker and as a repository.

When functioning as a broker, the TAXII Server sends information between systems. This is an interaction where one client sends a message to a TAXII Server, and the TAXII Server then delivers the message to other clients as appropriate.

When functioning as a repository, the TAXII Server responds to client requests directly. This is an interaction where a client makes a request to a TAXII Server, and the TAXII Server immediately fulfills the request with information available to the TAXII Server (nominally a database).

A compliant TAXII server may implement either the broker API portion of the specification, or the repository API portion of the specification, or both portions.

All client/server interactions in TAXII 2 follow the RESTful architectural pattern and use JSON over HTTPS.

Broker Concepts

The broker portion of TAXII is defined as containing a set of **Channels**. In TAXII, a Channel is a named construct that is used to transmit messages from a producer to consumers. Producers write messages to Channels; consumers read messages from Channels. Producers and Consumers are decoupled because they do not need to know about each other in order to communicate with each other - they only need to write to and read from the same Channel.

The TAXII Server is required to perform delivery activities for messages written to a Channel, but is not required to process messages on a Channel. Messages are processed by the writers and readers - the producers and consumers - of messages. TAXII Server implementations, as a value-add, may choose to package certain types of clients with a TAXII Server product. That packaging is outside the scope of this specification.

The TAXII API, defined later in this document, specifies the requirements for Channels.

Repository Concepts

The repository portion of TAXII is defined as containing a set of **Repositories** (this apparent redundancy will be cleared up shortly). In TAXII, a Repository is a storage and retrieval system for cyber threat intelligence. Since there are multiple formats for cyber threat intelligence, the Repository portion of the TAXII API permits the presence of multiple repositories - perhaps one repository for each type of threat intelligence that the TAXII Server contains or can produce.

The TAXII API, defined later in this document, specifies the requirements for Repositories.

TAXII API

This section contains normative requirements for the TAXII API. The TAXII API is designed to follow the RESTful architectural pattern and uses JSON exclusively. This section covers both broker and repository functionality.

HTTPS

All TAXII 2.0 RESTful services are over HTTPS

DNS SRV

DNS Service (SRV) records are used for advertising network services via DNS. This section defines a service name for TAXII so that TAXII implementations that conform to this specification can be advertised in DNS. This section also defines rules for using the TAXII DNS SRV record to determine the Discovery URL and the Default API Base.

The service name for this version of TAXII is: **“taxii2.”** Future versions of TAXII may define alternate service names.

An example DNS SRV record, advertising a TAXII server located at taxii.example.com:443:

`_taxii2._tcp.example.com. 86400 IN SRV 0 5 443 taxii.example.com`

Ref: https://www.ietf.org/rfc/rfc2782.txt

Default API Base:

TAXII defines the concept of a default API base. TAXII Servers that are advertised in DNS MUST have an API instance located at the Default API Base. The Default API Base also serves as the base for the optional Discovery URL.

The Discovery URL will enable the discovery of TAXII Brokers and or TAXII Repository services on this server or other servers. A detailed description of the REST API for this resource can be found in the REST API Section. This service MAY require authentication and MAY limit the results based on that authentication.

The Default API Base is created from the TAXII DNS SRV record using the following syntax:

Default API Base: `https://` + `hostname` + `:` + `port` + `/taxii2/`

Applying these rules to the example DNS SRV record results in the following Default API Base URL: https://taxii.example.com:443/taxii2/

API Base

This specification defines the concept of an API Base. An API Base is a URL that is used as the “root” URL for any particular instance of the TAXII API. TAXII Servers MUST have at least one API Base. The API Base concept enables a single TAXII Server to simultaneously host multiple instances of the TAXII API, should that feature be desired by the implementer. This specification does not define how API Bases are managed (i.e., created, modified, or deleted).

An example API Base: https://subdomain.example.com:12345/trustgroup1/

This specification uses the notation **[API-Base]** to refer to a generic API Base.

URL Reference

The following table provides a summary of TAXII API URLs. This section is an informative summary of normative requirements contained elsewhere in this document.

The table headings are defined as follows:

- **Path** - Indicates the URL that the HTTP request specifies
- **Method** - Specifies the HTTP Method (e.g., GET, POST) of the HTTP request.
- **Request Message Type** - Specifies entity-body of the HTTP Request
- **Success Response** - Identifies the HTTP Status Code and entity-body of a successful interaction. HTTP Status Codes for errors are defined ##REF##.

Path	Method	Request Message Type	Success Response
[API-Base]	GET	n/a	HTTP 200 Server Information
[API-Base]/channels/	GET	n/a	HTTP 200 Channel Listing
[API-Base]/channels/	POST	Channel Create	HTTP 201 Channel Info
[API-Base]/channels/<channel-name>/	GET	n/a	HTTP 200 TAXII Message(s)

			or HTTP 204 No New
[API-Base]/channels/<channel-name>/	POST	TAXII Message	HTTP 202 <TBD>
[API-Base]/repos/	GET	n/a	HTTP 200 Repository Listing
[API-Base]/repos/<repo-name>/	GET	n/a	HTTP 200 Repository Information

HTTP Status Code Reference

TAXII API uses a uniform set of HTTP status codes which are summarized in the following table. This section is an informative summary of normative requirements contained elsewhere in this document.

HTTP Status Code	**Meaning** (From HTTP RFC)	**Used When**
HTTP 200	OK	The client's request succeeded and there was information available
HTTP 201	Created	The client's request to create a resource succeeded.
HTTP 202	Accepted	The client submitted a message to a Channel and the message was accepted.
HTTP 204	No Content	The client's request succeeded but there was no information available
HTTP 400	Bad Request	The server cannot understand the request
HTTP 403	Permission Denied	The client is not authorized to make the request against the requested resource
HTTP 501	Not Implemented	The client has requested a

		function that is not implemented on the server

HTTP X-Headers

This specification defines X-Headers that clients can use to request certain behaviors from the server they are connecting to. All headers defined in this section are optional for clients to use and required for servers to support.

X-Header Name	Description	Allowable Values	
X-Max-Size	The `X-Max-Size` header identifies the maxmimum response size in decimal number of OCTETs that the client is capable of receiving. The `Content-Length` of the response MUST NOT be larger than the `X-Max-Size` value, if specified.	The mimimum allowable value is `9437184` (9mb). There is no maximum allowable value.	

URL Requirements

This section contains normative requirements for all TAXII API URLs, including which HTTP verbs (e.g., GET, POST) are permitted, which messages types are used, and which HTTP Status Codes should be used.

This specification attempts to leverage “vanilla HTTP” as much as possible, and therefore gently reminds readers of the following:

1. HTTP 200 (OK) is used for successful requests
2. HTTP 201 (Created) is used when the requests has succeeded and a new resource has been created.
3. HTTP 202 (Accepted) is used when the request has been accepted but not processed.

4. HTTP 204 (No Content) is used when the request has been fulfilled but there is no content to return.
5. HTTP 400 (Bad Request) is used for malformed requests
6. HTTP 401 (Unauthorized) is used for requests where authentication is required
7. HTTP 403 (Forbidden) is used when the request does not have sufficient permissions to perform the request
8. HTTP 405 (Method Not Allowed) is used when the request uses a method (e.g., TRACK) that is not permitted.
9. HTTP 406 (Unacceptable) is used when the request's Accept header specifies a formatting the server cannot produce for the identified resource (i.e.,URL).
10. HTTP 415 (Unsupported Media Type) is used when the request's Content-Type header is not supported by the server for the identified resource (i.e., URL).
11. HTTP 501 (Not Implemented) is used when the server does not support the functionality needed to fill the request.

[API-Base]

This URL is the root of a TAXII API and therefore permits retrieval of information about the instance of the TAXII API instance it anchors. The **[API-Base]** URL does not provide any facility for actually exchanging cyber threat information - it instead provides meta-information about the TAXII Server it resides on and **[API-Base].**

HTTP Method	Response Message	HTTP Status Code	Description
GET	API-Base Information	HTTP 200 - OK	Returns information about the TAXII Server hosting this **[API-Base]** and this **[API-Base]**.

[API-Base]/channels/

This URL represents, for a particular TAXII API instance, the entry point into the Channels portion of the API. Channels can be created, destroyed, modified, and listed by performing actions against this URL.

HTTP Method	Response Message	HTTP Status Code	Description

GET	Channel Listing	HTTP 200 - OK	Returns information about the channels in this instance of the TAXII API.
POST			Create a Channel
PUT			Create/Modify a channel
DELETE			Delete a Channel

Messages

This section contains messages used in TAXII. Each message definition includes a table defining the message and an example message.

Plus signs are used to indicate parent/child because split cells isn't a feature of google docs.

TODO: What MIME type is used for these messages?

API-Base Information

Field Name	Description	JSON Type	Allowable Values
type	A static field identifying the type of this object	string	Only the literal 'api-base-information' is permitted
contact-info	A text field containing human-orented contact information	string	
description	A text field containing a human-oriented description of the TAXII Server	string	

Channel Listing

Field Name	Description	JSON Type	Allowable Values
type	A static field identifying the type of this object	string	Only the literal ‘channel-listing’ is permitted
channels	A list of channel information objects	array	The array MUST contain only channel information objects
Channel Information object			
type	A static field identifying the type of this object	string	Only the literal ‘channel-information’ is permitted
name	The name of the channel	string	tbd
url	The URL of the channel	string	tbd
description	A prose description of the channel	string	tbd
subscribers	A list of subscriber IDs	array	The array MUST contain strings that conform to <???>
can-read	Indicates whether the requestor can read from the channel	boolean	True/False
can-write	Indicates whether the requestor can write to the channel	boolean	True/False


```
{
  "type": "channel-listing",
  "channels": [
    {
      "type": "channel-information",
      "name": "channel 1",
      "url": "https://example.com/taxii/channels/channel-1/",
```

```
      "description": "This is a description. Read it.",
      "subscribers": ["mdavidson", "bjordan"],
      "can-read": True,
      "can-write": False
    }
  ]
}
```

TAXII Messages (?)

TBD

Repository Listing

Field Name	Description	JSON Type	Allowable Values
type	A static field identifying the type of this object	string	Only the literal ‘repository-listing’ is permitted
channels	A list of Repository Information objects	array	The array MUST contain only Repository Information objects
Repository Information object			
type	A static field identifying the type of this object	string	Only the literal ‘repository-informatio n’ is permitted
???			

Messaging Considerations

This section defines requirements for how TAXII Servers deliver messages.

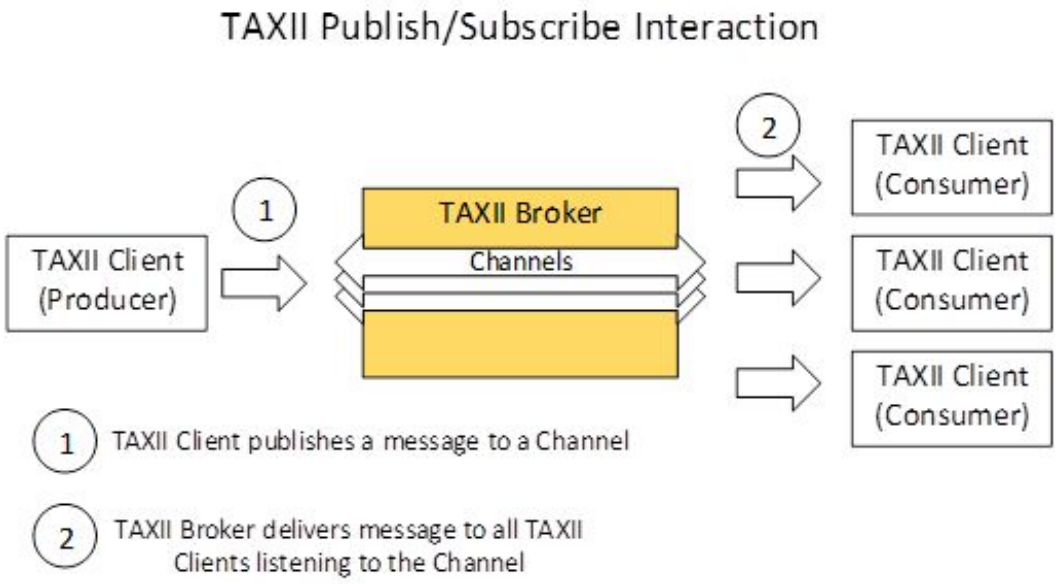
Topics to include:

- Guaranteed deliver (MSD:

BELOW HERE IS NOT PART OF THE PRE-DRAFT AND IS JUST TEXT THAT IS KEPT FOR FUTURE REFERENCE AND WILL BE DELETED LATER

TAXII Broker

This section describes and defines a TAXII Broker. A TAXII Broker has Channels, which are used to transmit messages (often containing cyber threat information) from a producer to one or more consumers. A rough approximation of this concept is “email for applications”; those familiar with messaging technologies will recognize this as the publish-subscribe messaging pattern. Note that this section is descriptive in nature and does not contain any normative statements.



TAXII Brokers facilitate the following interactions with clients:

Channel Creation:

1. A TAXII Client sends a “Channel Create” message to the TAXII Broker
2. The TAXII Broker responds with a message indicating the success or failure of the request

Channel Deletion

1. A TAXII Client sends a “Channel Delete” message to the TAXII Broker

2. The TAXII Broker responds with a message indicating the success or failure of the request

Message Production:

1. A TAXII Client sends a message to a specific channel
2. The TAXII Broker responds with a message indicating the success or failure of the request

Message Consumption:

1. The TAXII Client sends a message establishing a subscription on a channel and requests messages from that channel in accordance with the subscription
2. The TAXII Broker responds with available messages, an error condition, or a response indicating that while the request succeeded there were no messages available to be sent.

A summary of the broker concept:

- A TAXII Broker contains channels
- Producers send messages to channels
- Consumers receive messages from channels
- Producers and consumers do not interact directly

TAXII Server

A TAXII server includes a TAXII Broker Service and a TAXII Repository Service. DNS Service Resource records MAY be used to help in the autodiscovery of your TAXII Server.

HTTPS

This specification defines requirements for using HTTPS; this specification does not define requirements for using HTTP. TAXII Servers MUST use HTTPS. While the additional use of HTTP is recognized as a straightforward and perhaps trivial to implement, and while Clients and Servers may choose to use HTTP, those uses are not considered conformant with this specification. ONLY the use of HTTPS is considered conformant with this specification.

Authentication

TAXII Services use JWT (JSON Web Tokens) as a default authentication method. Additional authentication services and multi-factor authentication MAY be added.

Error codes for authentication are: (TODO)

HTTP 201 - ??
HTTP 400 - ??
HTTP 403 - ??
HTTP 501 - ??

TAXII Repository Service

This section hasn't really been fleshed out yet. See Trey Darley's proposal.
http://taxiiproject.github.io/taxii2/notional-query-api/

RESTful API

This section defines the TAXII Server, TAXII Broker, and TAXII Resource RESTful APIs

TAXII Server API

The defined default Discovery URL is created from the TAXII DNS SRV record using the following syntax:

Discovery URL: https:// + hostname + : + port + /taxii2/api/
Example Discovery URL: https://taxii.example.com:443/taxii2/api/

API-Base

Note that in this version of TAXII, the creation, modification, and deletion of API-Bases is not currently specified, but may be added in a future version of TAXII; only the listing of available API-Bases is specified. Implementers are free to implement (or not) create/modify/delete functionality in any way they wish.

An API-Base is any valid HTTPS URL (e.g., https://subdomain.example.com:12345/whatever/)

The following HTTP verbs are supported for the /taxii2/api/. In a future version additional verbs may be added.

- GET:

Error codes for this resource are:
HTTP 201 - Message accepted, discovery information will be provided

HTTP 400 - Invalid request, something was wrong with your request
HTTP 403 - Permission denied, you are not allowed to query the discovery service
HTTP 501 - Not implemented, this server has not implemented the discovery service

A GET on the /taxii2/api/ resource will return a JSON structure with the following fields.

- “type” = string - message type with a value of “discovery”
- “auth” = strings - supported authentication type (TODO, may need to make this an array, need to figure out how to define implementation specific or layered multi-factor auth types)
- “brokers” = array of strings - where each string is a full URI
- “repos” = array of strings - where each string is a full URI

Example

```
{
    "type": "discovery",
    "auth": "jwt",
    "brokers": [
        "https://taxii-foo.example.com:443/taxii2/broker1/",
        "https://taxii-bar.example.com/t2/broker2/"
    ],
    "repos": [
        "https://taxii-foo.example.com:443/taxii2/repo1/",
        "https://taxii-bar.example.com/t2/repo2/"
    ]
}
```

TAXII Broker API

The default TAXII Broker API Base is an API entry point if no other API Base is defined. While it is possible to delete this default entry point, each TAXII Broker service should include this by default, out of the box.

The Default TAXII Broker API Base is created from the TAXII DNS SRV record using the following production:

Default API Base: https:// + hostname + : + port + /taxii2/broker/
Example Default API Base: https://taxii.example.com:443/taxii2/broker/

For a TAXII Broker Service, the following URLs are valid:

[API-Base]/channels/<channel-name>/

[API-Base]/channels/

Broker API Methods

In this section, each URL is specified, along with a set of possible responses.

GET Channels

A GET on the [API-Base]/channels/ resource will return a list of channels on this TAXII Broker as a JSON structure with the following fields

- “type” = string - message type with a value of “channels”
- “channels” = array of strings - where each string is a channel

Example

```
{
    "type": "channels",
    "channels": [
        "indicator",
        "malware"
    ]
}
```

Error codes for this resource are:
HTTP 200 - Returns a list of channels accessible by the requestor (e.g., channel-list message)
HTTP 400 - Invalid request
HTTP 403 - Permission denied, you are not allowed to view the channel list

Get Channel <name>

A GET on the [API-Base]/channels/<channel-name>/ resource will return information about a specific channel as a JSON structure with the following fields

- “type” = string - message type with a value of “channel-details”
- “name” = string - name of channel
- “long-poll” = bool - does this channel make use of HTTP long polling
- “msg-age” = string - how long in seconds does the server hold messages for clients before purging them

Example

```
{
    "type": "channel-details",
    "name": "<channel-name>",
    "long-poll": true,
```

```
    "msg-age": "86400"
}
```

Error codes for this resource are:
HTTP 200 - Returns channel information (e.g., channel-info message)
HTTP 400 - Invalid request
HTTP 403 - Permission denied, you are not allowed to view this channel's information

POST <channel object> to [API-Base]/channels/ - Create a new channel
HTTP 201 - Message accepted, channel will be created
HTTP 400 - Invalid request, something wrong with your channel structure
HTTP 403 - Permission denied, you are not allowed to make new channels
HTTP 501 - Not implemented, this server has a fixed set of available channels

PUT <channel object> to [API-Base]/channels/<channel-name>/ - Edit / update details of a channel

POST <taxii message(s)> to /channels/<channel-name>/ - Add TAXII message(s) to the channel
HTTP 202 - Message(s) accepted, message will be transmitted to the channel
HTTP 400 - Invalid request, something wrong with your message structure
HTTP 403 - Permission denied, you can not post to this channel
HTTP 501 - Not implemented (This is a read only channel)

GET /channels/<channel-name>/[?param1=val1...] - Get or Create a subscription and get TAXII messages from the channel with the specified filter. For more on subscriptions, see below.
HTTP 200 - Response fulfilled
HTTP 204 - No new data
HTTP 400 - Invalid request
HTTP 403 - Permission denied
HTTP 501 - Not implemented (This is a write-only channel)

TAXII Repository API