TAXII 2.0 Specification Pre-Draft

Version 0.2

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.

This specification addresses a number of key known issues with TAXII 1.1:

- Lack of automated discovery; Addressed by defining a DNS SRV record
- Lack of real-time exchanges; Addressed by HTTP Long Polling
- Lack of specificity regarding authentication; Addressed by defining authentication requirements
- Too much optionality (a.k.a., lack of a single architecture); Addressed by more concretely defining what a TAXII Server “is” and how it should behave.

TODO: Explain that services went away
TODO: Explain TAXII 1.x Data Collection vs. 2.x Collection

Similarities and Differences

Table of Contents

Section 1: TAXII Overview

TAXII enables communication of cyber threat information across product and organizational boundaries. This specification defines the TAXII Protocol and requirements for TAXII Server implementations.

TAXII Protocol Overview

The TAXII Protocol follows the RESTful architectural pattern and uses JSON/HTTPS for all communications. This specification defines the URL Structures, JSON Messages, and authentication methods that TAXII Servers are required to implement. The TAXII Protocol is “HTTPS only”, meaning that communications over HTTP are not conformant with this specification.

TAXII Server Overview

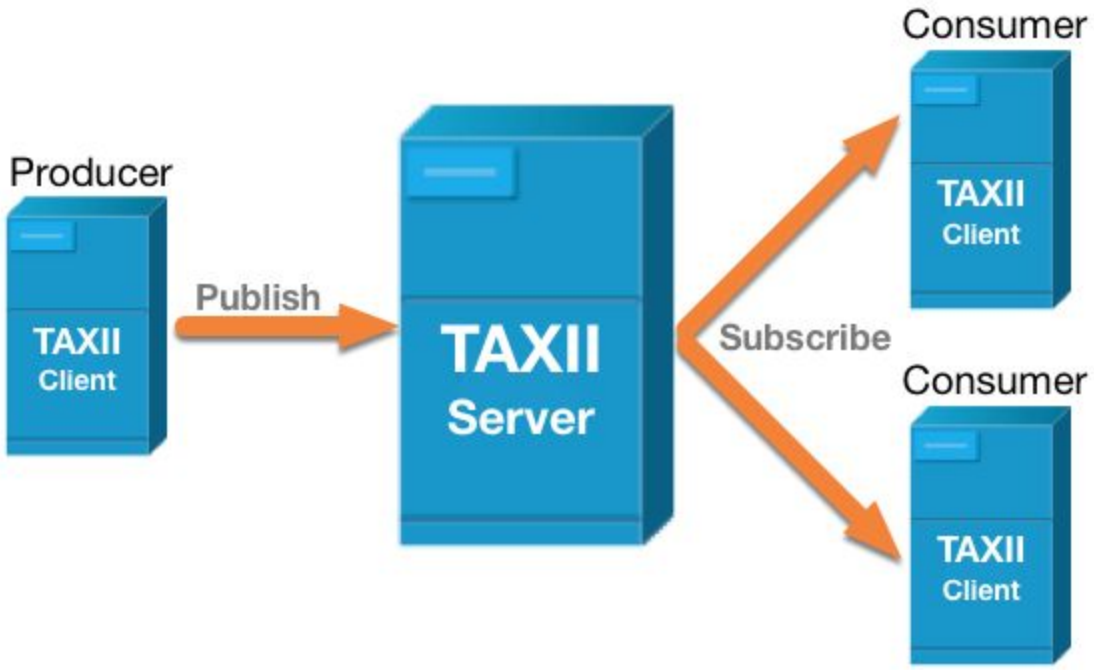
A TAXII Server is a piece of cyber security infrastructure that facilitates the exchange of threat information using Channels and Collections.

Channels Overview

A **Channel** distributes information from producers to consumers. Producers send information to a Channel and Consumers receive information from a Channel. Channels are used to exchange information in an asynchronous, event-based manner where both Producers and Consumers are clients and the Channel is maintained by the server.

TAXII Servers perform delivery activities for information written to a Channel; information is processed by Producers and Consumers. TAXII Server implementations, as a value-add, may choose to add policy based processing of messages on a channel or package certain types of clients with a TAXII Server product. Those options are outside the scope of this specification.

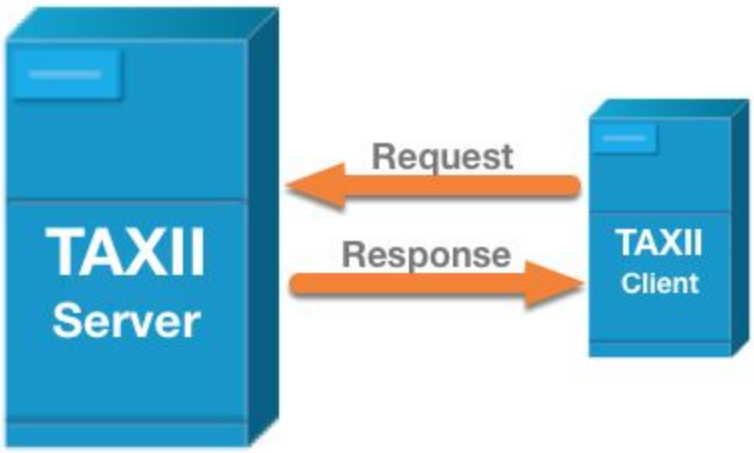
The diagram below illustrates a single producer sending a message to the TAXII Server, and that TAXII Server then distributing the message to multiple consumers. Normative requirements for Channels are defined later in this document.



Collections Overview

A **Collection** is used by clients to either send information to the server or request information from the server. A TAXII Server may contain multiple Collections and Collections are used to exchange information in a request-response manner. The server responds to client requests directly, and only the client and server see the request-response communication.

The diagram below illustrates an interaction where a client makes a request to a TAXII Server, and the TAXII Server fulfills the request with information available to the TAXII Server (nominally from a database). Normative requirements for Collections are defined later in this document.



DNS Service (SRV) Records

This specification defines a DNS SRV record to allow advertisement of TAXII Servers, if desired, via DNS. Using this feature will enable autodiscovery of a TAXII Server and its features.

Authentication

This specification defines a mandatory to implement authentication method of HTTP Basic with JSON Web Tokens (JWT). As all communications in TAXII are over HTTPS, HTTP Basic is an acceptable authentication system to guarantee baseline interoperability. This specification will also define extension points for other authentication systems to accommodate the diverse needs within TAXII deployments. For more information and details about using authentication with TAXII and enabling authentication systems other than HTTP Basic with JWT, please see the section titled, Authentication.

Section 2: TITLE TBD

HTTP

This specification use the HTTP protocol for all communications and requires that all TAXII servers **MUST** use HTTPS [RFC7230] and implement TLS version 1.2 [RFC5242]. Further, TAXII servers **MUST** implement TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 [RFC5489].

With the exception of TLS_ECDHE_RSA_WITH_NULL_SHA384 and the NULL cipher suites, a TAXII server **MUST NOT** offer or negotiate (bid down) an encrypted connection with parameters weaker than TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.

TAXII clients **MUST** use HTTPS [RFC7230] and implement TLS 1.2 [RFC5242]. TAXII clients can expect conforming TAXII servers to implement at least TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 [RFC5489]. With the exception of TLS_ECDHE_RSA_WITH_NULL_SHA384 and the NULL cipher suites, a TAXII client **MUST NOT** offer or negotiate (bid down) an encrypted connection with parameters weaker than TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.

Operators will deploy TAXII clients and servers per their local security policy. The requirements for a TAXII server to implement HTTPS, TLS 1.2, and TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 set a minimum baseline for servers such that in any environment sufficient security is available to clients that desire it. From an interoperability perspective, even if a particular deployment might not care about any security or integrity or privacy at all, and as such would be happy to have clients and servers totally vulnerable to man-in-the-middle attacks and privacy leakage, there will be many deployments that will require the minimal level of security offered by HTTPS/TLS 1.2/ECC-AES-SHA-2. As such, if a “TAXII server” does not implement this baseline, it cannot call itself a ‘TAXII’ server. Likewise, a TAXII client that fails to implement TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 may find itself unable to communicate with a TAXII server.

If local policy at the TAXII server is to allow transfers in the clear, in the TLS handshake the server can offer TLS_ECDHE_RSA_WITH_NULL_SHA384 for integrity checked transfers or the NULL cipher for insecure transfers. If local policy at the TAXII client is to allow transfers in the clear, the client can also offer in the TLS handshake TLS_ECDHE_RSA_WITH_NULL_SHA384 for integrity checked transfers or the NULL cipher for insecure transfers. Normal TLS negotiation will result in the appropriate cipher suite per the mutually agreed to security level.

We expect many deployments will have more stringent security policies. For example, this TAXII specification is silent on certificate authorities. For many uses, the Web model of having hundreds of root certificate authorities may be sufficient. For other users, a single or at most a pair of root certificate authorities may be mandated. For other users, a self-signed certificate may be sufficient. All of these policies are up to the organizations deploying the TAXII clients and servers. This specification allows for the range of totally open (hundreds of 'recognized' root certificate authorities) to totally closed (only a single, either hard-coded, physically distributed, or limited set of potentially pinned certificate authority, perhaps self-signed by a closed user community).

Likewise, this TAXII specification is silent as to validation and revocation policy. This is again a policy issue. We would expect most TAXII servers to implement the X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP) [RFC6960]. However, that is a policy issue, not a mandate for implementation.

DNS SRV Records

This specification defines an optional DNS Service (SRV) record [RFC2782] for use with TAXII. The service name for this version of TAXII is: "taxii". Future versions of TAXII may define alternate service names.

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

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

Long Polling

***TAXII does long polling. Need to specify things related to long polling here.
TODO This needs to be fleshed out ***

Section 3: TAXII API

The TAXII API is designed as a resource-oriented HTTP interface that uses JSON. The TAXII API uses the URL production “/<resource-type>/<resource-identifier>/” and leverages native HTTP semantics (e.g., HTTP Status Codes, MIME types).

Discovery API

This specification defines a single Discovery API URL that clients can use, if implemented, to discover the TAXII capabilities that the server offers as well as meta-information about the TAXII Server (e.g., contact information). When this Discovery API is used with a DNS SRV record, clients can auto-discover TAXII services as follows:

1. Client uses DNS to retrieve a TAXII DNS Service Record
2. Client uses the TAXII DNS Service Record to construct the Discovery API URL
3. Client issues a request to the Discovery URL

Recall the example TAXII DNS Service record:

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

Using port (443) and the hostname (taxii.example.com) from the DNS Service Record, the TAXII Discovery URL is constructed as follows:

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

A fully constructed TAXII Discovery URL is as follows:

`https://taxii.example.com:443/taxii/`

For backward compatibility with previous versions of TAXII, implementations **MAY** advertise previous versions of TAXII in the Discovery message. This specification uses the notation **[Discovery]** to refer to the Discovery API URL.

API Base

This specification defines an API Base 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. A single TAXII Server **MAY** host as many instances of the TAXII API as desired, with each instance located at a unique API Base. Hosting multiple API Bases would allow an implementer to mimic trust

groups or groups of interest on a single TAXII Server. Different API Bases **MAY** have different authentication requirements. This specification does not define how API Bases are managed (i.e., created, modified, or deleted). This specification uses the notation **[API-Base]** to refer to a generic API Base.

An example API Base:

`https://subdomain.example.com:12345/trustgroup1/`

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.

HTTP Status Codes

TAXII 2.0 uses HTTP Status Codes to communicate status information about requests. This section provides a summary of HTTP Status Codes that implementers need to consider. This specification defines no requirements for the use of HTTP Status Codes.

Status Code	Meaning (From HTTP RFC)	Is Used When …
HTTP 200	OK	… the request succeeded
HTTP 201	Created	… the request succeeded AND a new resource (e.g., a Channel) was created

HTTP 202	Accepted	… the request has been accepted but not processed
HTTP 204	No Content	… the request has been fulfilled and there is not content to return
HTTP 400	Bad Request	.. the server cannot understand the request due to malformed syntax
HTTP 401	Unauthorized	… authentication is required
HTTP 403	Forbidden	… the request could not be fulfilled due to lack of authorization
HTTP 405	Method Not Allowed	… the request specifies an HTTP verb (e.g., POST) that is not permitted.
HTTP 406	Unacceptable	… the request's Accept header does not specify a value the server can supply for the specified resource (URL)
HTTP 415	Unsupported Media Type	… the request's Content-Type header is not supported by the server for the specified resource (URL)
HTTP 501	Not Implemented	… the client has requested a function that is not implemented on the server

URL Summary

This section summarizes the URLs defined in the TAXII API. Full definitions and requirements for each URL are in subsequent sections.

Path	Description
[Discovery]	Enables discovery of Server Metadata and TAXII functionality.
[API-Base]/	Enables discovery of resources (channels / collections) at this API Base
[API-Base]/channels/	Enables discovery and management of Channels.
[API-Base]/channels/<channel-name>/	Enables interacting directly with a specific Channel (e.g., reading, writing).
[API-Base]/collections/	Enables discovery and management of

	Collections.
[API-Base]/collections/<collection-name>/	Enables interacting directly with a specific Collection (e.g., creating or requesting resources).

The following 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** - Specifies the message type of the HTTP Request
- **Response Message** - Specifies the message type of the HTTP Response.

Path	Method	Request Message Type	Response Message Type
[Discovery]	GET	n/a	Discovery
[API-Base]	GET	n/a	API Status
[API-Base]/channels/[?params]	GET	n/a	Channel Listing
[API-Base]/channels/	POST	Channel Create	Channel Status
[API-Base]/channels/<channel-name>/	GET	n/a	TAXII Data
[API-Base]/channels/<channel-name>/	POST	TAXII Data	TAXII Status
[API-Base]/collections/[?params]	GET	n/a	Collections
[API-Base]/collections/	POST	Collection Create	Collection Status
[API-Base]/collections/<collection-name>[?params]	GET	n/a	TAXII Data
[API-Base]/collections/<collection-name>/	POST	Object Create	<TBD>

URL Details

This section defines the behavior of each URL in the TAXII API. Note that for the sake of brevity, error related messages flows are not described in each URL's section. Generally, each channel has a general description and a listing of message flows.

[Discovery]

This URL is the root of a TAXII Server and permits retrieval of server-wide information and API Bases running on this or other servers.

[API-Base]

This URL is the root of a TAXII API and permits retrieval of meta-information about this instance of the TAXII API. The **[API-Base]** URL does not provide any facility for actually exchanging cyber threat information**.**

The following table illustrates the supported HTTP methods at this URL.

HTTP Method	Response Message	HTTP Status Code	Description
GET	API-Status	HTTP 200 - OK	Returns information about the channels in this instance of the TAXII API.

[API-Base]/channels/

This URL contains all Channel resources within the **[API-Base],** enabling discovery and management of these Channel resources. GET requests to this URL will result in a listing of available Channel resources (possibly filtered by query parameters or by policy). POST requests to this URL will attempt to create a new Channel resource or modify an existing Channel resource. Requests and responses that are permitted to contain message bodies MUST be able to be represented as a MIME-type of application/taxii+json.

Requests to this URL **MUST NOT** result in an HTTP 405 (Unacceptable) when the ##TODO# is specified in the Accept header.

GET Requests

GET requests to this URL are used to request a listing of available Channel resources, possibly filtered by query parameters. HTTP GET requests to this URL that result in an HTTP 200 response **MUST** contain a Channel Listing message (even if the Channel Listing message does not list any Channels) in an acceptable format (per the HTTP Accept header in the request).

HTTP GET requests **MAY** contain any combination of the following query parameters:

Parameter	Description	Allowable Values

name	Specifies the name of the Channel being requested. Exact matches are required.	TBD

POST Requests
HTTP POST requests to this URL **MUST NOT** result in an HTTP 415 when the Content-Type is ##TODO## and the content is valid per the specified Content-Type.

This URL has four message flows.

Channel Creation
This message flow creates a channel.

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

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

GET http 200
or
GET http 204 no new
POST http 202

DELETE Requests
TAXII Servers **MUST** be able to respond to HTTP DELETE requests to this URL and **MUST** support the following query parameters:

- name - channel name

Channel Names **MUST** have a length from 1 to 128 characters long and only contain upper and lower case ascii characters (A-Z and a-z), numbers (0-9), and a hyphen character (-) in accordance with the following regular expression: `^[A-Za-z0-9-]{1,128}$`

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)

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

- “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"
```

}

[API-Base]/collections/

GET HTTP 200

[API-Base]/collections/<collection-name>/

GET HTTP 200

Collection Names **MUST** have a length from 1 to 128 characters long and only contain upper and lower case ascii characters (A-Z and a-z), numbers (0-9), and a hyphen character (-) in accordance with the following regular expression: ^[A-Za-z0-9-]{1,128}$

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

Section 4: 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?

Discovery

Field Name	Description	JSON Type	Allowable Values
type	A static field identifying the type of this object	string	Only the literal 'discovery' 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	
api-bases	An array of string values that	array	

	contain api-bases that this servers knows about		

Example

```
{
    "type": "discovery",
    "contact-info": "Bret and Mark at company xyz",
    "description": "This server support the following company xyz",
    "api-bases": [
        "https://taxii-foo.example.com:443/taxii2/collection-foo/",
        "https://taxii-bar.example.com/t2/channel-bar/"
    ]
}
```

API-Status

Field Name	Description	JSON Type	Allowable Values
type	A static field identifying the type of this object	string	Only the literal ‘api-status’ 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
    }
  ]
}
```

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-information’ is permitted
???			

Messaging Considerations

This section defines requirements for how TAXII Servers deliver messages.

Topics to include:

- Guaranteed deliver (MSD:

Section 5: Authentication

Error codes for authentication are: (TODO)
HTTP 201 - ??
HTTP 400 - ??
HTTP 403 - ??
HTTP 501 - ??

Section 6: Policy Based Authorization