TAXII™ 2.1 Interoperability Test Document Version 1.0

Working Draft 01

01 March 2022

Technical Committee:
[OASIS Cyber Threat Intelligence (CTI) TC](#)

Chairs:
Robert Coderre (robert.c.coderre@accenture.com), Accenture
Trey Darley (trey.darley@cert.be), CCB/CERT.be

Editors:
Bret Jordan (bj@ctin.us), Cyber Threat Intelligence Network, Inc.
Marlon Taylor (marlon.taylor@cisa.dhs.gov), DHS
Allan Thomson (athomson@lookingglasscyber.com), LookingGlass
Jason Keirstead (jason.keirstead@ca.ibm.com), IBM
Justin Stewart (jstewart@lookingglasscyber.com), LookingGlass
Dez Beck (dbeck@mitre.org), MITRE Corporation
Daniel Haynes (dhaynes@mitre.org), MITRE Corporation
Kartikey Desai (khdesai@mitre.org), MITRE Corporation

Related work:

This document is related to:

- *STIX™ Version 2.1*. Edited by Bret Jordan, Rich Piazza, and Trey Darley. http://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html
- *TAXII™ Version 2.1.* Edited by Bret Jordan and Drew Varner. Latest stage: https://docs.oasis-open.org/cti/taxii/v2.1/taxii-v2.1.html
- *STIX™ Interoperability Test Document Version 1.0.* Edited by John-Mark Gurney, Bret Jordan, Michael Rosa, Marlon Taylor, Rajesh Patil, Justin Stewart, and Kartikey Desai. https://docs.google.com/document/d/1SabxIhjxfg1RAaBj6grsktBzX_UHFI4C8VYStybi-c0/edit

Abstract:

This is the Interoperability test document to supplement the Trusted Automated Exchange of Intelligence Information 2.1 OASIS Standard developed by the Cyber Threat Intelligence Technical Committee (CTI TC) of the Organization for the Advancement of Structured Information Systems (OASIS). This test document provides detailed requirements on how product implementers within the threat intelligence ecosystem may demonstrate TAXII 2.1 interoperability compliance. There are several personas detailed in section 1.3 of this document. These are: TAXII Client (TXC) and TAXII Server (TXS). This Interoperability test document defines tests of the following use cases: authentication and authorization, server discovery, GET API Root information, GET collections, GET a collection, GET object manifests, GET objects, GET an object, GET object versions, add objects, GET status, DELETE an object, filter

results, pagination, and custom properties. For each of these use cases the document defines what the TXC and TXS need to support to satisfy each test case.

Status:

Citation format:

When referencing this document the following citation format should be used:

[CitationLabel]

##specname## Version ##docver##. Edited by Patrick Durusau and Michael Brauer. ##docdate##. OASIS ##docstage## ##docrev##. https://docs.oasis-open.org/office/v1.2/csd07/OpenDocument-v1.2-csd07.html. Latest version: https://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2.html.

(VISIBLE hyperlink to HTML version) (replace example above with citation for document)

(CitationLabel is short text based on Title and Version, not stage.)

Basic template is:

Work Product title (italicized). Edited by Albert Alston, Bob Ballston, and Calvin Carlson. Approval date (DD Month YYYY). OASIS Stage Identifier and Revision Number (e.g., OASIS Committee Specification Draft 01). Principal URI (version-specific URI, e.g., with filename component: somespec-v1.0-csd01.html). Latest version URI (i.e., without stage components).

Notices

Table of Contents

1 Introduction

This document provides details of the Trusted Automated Exchange of Intelligence Information (TAXII) 2.1 Interoperability Test Document. It lists a set of use cases that a persona (see section 1.3) **MUST** follow as they develop minimally viable TAXII-compliant tools and services. To claim TAXII interoperability compliance, persona tools/services **MUST** adhere to expected behaviors and outcomes as detailed in the use cases.

The OASIS Cyber Threat Intelligence Technical Committee (CTI TC) recommends users of this test document become familiar with the TAXII 2.1 OASIS Standard https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.html (as given in the Related Work section above) prior to implementing the use cases in this document. This is what this document is referring to when it mentions "TAXII 2.1 OASIS Standard".

NOTE: The TAXII 2.1 OASIS Standard contains normative references to other specifications with which an implementation **MAY** need to reference and meet in order to comply with these specifications. This document assumes that such requirements are also met.

1.1 Terminology

Client - A software instance that can connect to and utilize the services/resources of a server
Server - A software instance that enables and manages access to a resource or service
TAXII container resource - Either a TAXII Envelope, Manifest Resource, or Versions Resource

1.2 Overview

The approach that is being taken within the CTI TC is to rely primarily on well-defined, common use cases to drive the interoperability between products using TAXII 2.1. Section 3 of this document outlines these common use cases for organizations seeking to develop and demonstrate interoperability.

These use cases will enable personas (see section 1.3) of the cyber threat intelligence information sharing community to build and test information sharing systems that are compliant with TAXII 2.1 interoperability. Future revisions to the TAXII 2.1 OASIS Standard will be incorporated into a new version of this document.

1.3 Personas

For an organization to demonstrate TAXII 2.1 interoperability compliance, their software instances will adhere to persona behavior and prescribed content as detailed in the test cases.

For documenting interoperability compliance for each persona tested, refer to the checklist and test requirements in section 4 Persona Checklist of this document. The following system personas are used throughout this document.

- TAXII Client (`TXC`) - A software package that connects to a TAXII Server and supports the exchange of CTI.
- TAXII Server (`TXS`) - A software package that supports the exchange of CTI.

2 Use Case Details

This Test Document defines a set of interoperability requirements for each persona defined in section 1.3. All use cases require the use of a TAXII Server (`TXS`) in concert with the TAXII Client (`TXC`) persona components as shown below.

A software instance **MAY** implement multiple personas. Therefore, it is conceivable that a single software instance **MAY** support both the TXC and TXS personas. However, for the purposes of this test case document, each persona's required behavior is called out separately.

The following figure provides a simplified diagram to highlight the relationship between a TXC and a TXS. In some cyber threat intelligence sharing ecosystems, TAXII Servers can support multiple TAXII Clients and TAXII Clients can support multiple TAXII Servers.

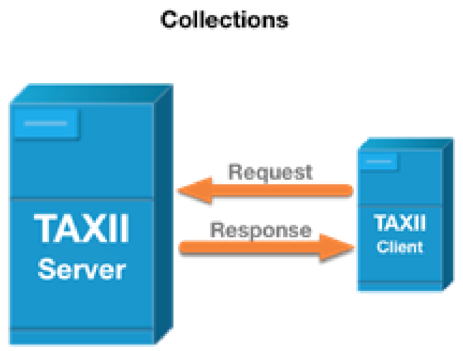

Figure 1. TAXII Client and TAXII Server Interactions

This document details the following use cases.

Table 1 - List of TAXII Interoperability Use Cases

	TAXII Client (TXC)	TAXII Server (TXS)
Authentication & Authorization	Required	Required
Certificate-based Authentication	Required	Optional
Server Discovery	Required	Required
GET API Root Information	Required	Required
GET Collections	Required	Required
GET a Collection	Required	Required
GET Object Manifests	Required	Required
GET Objects	Required	Required

GET an Object	Required	Required
GET Object Versions	Required	Required
Add (POST) Objects	Required	Required
GET Status	Required	Required
GET Status - All Status Properties	Required	Optional
DELETE an Object	Required	Required
Filter Results - TAXII 2.1 OASIS Standard	Required	Required
Filter Results - Additional Filters	Required	Required
Pagination	Required	Required
Custom Properties	Required	Required

2.1 Common Use Case Requirements

The following use case requirements apply to all tests in section 3.

2.1.1 Protocols

a. The HTTPS over IPv4 protocol **MUST** be used for all test cases.
b. There are no defined tests in this document that exclude IPv6 support.

2.1.2 Object Content

For the purposes of TAXII 2.1 Interoperability, all HTTP Accept and Content-Type headers will be TAXII version 2.1 media type "`application/taxii+json;version=2.1`". The TAXII Interoperability document will focus on using TAXII container resources (i.e., TAXII Envelope, Manifest Resource, and the Versions Resource).

In this document, TAXII Clients will use STIX 2.1 content and **MAY** conform to personas defined in section 1.2.1 of the STIX 2.1 Interoperability test document.

For the purposes of this TAXII Interoperability document, when a TXC sends objects to a TXS, the TXC **MUST** include all of the referenced objects within a single TAXII container resource. However, when a TXS is sending objects to a TXC, all of the objects **MAY** not necessarily be contained within a single TAXII container resource; see section 3.14 for more details.

2.1.3 Empty Lists

Section 2 of the TAXII 2.1 OASIS Standard, "Empty lists are prohibited in TAXII and **MUST NOT** be used as a substitute for omitting optional properties."

2.1.4 User-Agent Strings

A TXC **MUST** include its software name and version in the User-Agent HTTP header when transmitting a request to a TXS. See section 3.2 of the TAXII 2.1 OASIS Standard for more details.

A TXS **MUST NOT** reject a request that is missing the User-Agent header from a TAXII Client which conforms to section 8.4 of the TAXII 2.1 OASIS Standard, but does not conform to the requirements in 4.1 TAXII Client (TXC).

2.1.5 Custom Properties

A TXS or a TXC **MAY** encounter custom properties in the content it receives, and/or it **MAY** include custom properties in the content it sends. For more details on ensuring interoperability while handling custom properties, see section 3.15.

2.1.6 TLS Cipher Suites

A TXC **MUST NOT** use TLS 1.2 with any of the cipher suites that are listed in the cipher suite blacklist in Appendix A of [RFC7540].

2.1.7 Sorting

A TXS returning a Collections Endpoint response **MUST** sort Collection Resources in ascending order by **id**. See section 3.3 of the TAXII 2.1 OASIS Standard for more information about sorting.

2.2 Authentication and Authorization

The TAXII 2.1 OASIS Standard provides authentication and authorization schemes used by TXS and TXC. Please see section 1.6.9 and section 8 in the TAXII 2.1 OASIS Standard for further details.

TXS **MUST** implement support for at least one of the following authentication methods: HTTP Basic authentication (see section 8.2.2), certificate-based authentication (see section 8.3.1).

TXC **MUST** implement support for both HTTP Basic authentication (see section 8.5.1) and certificate-based authentication (see section 8.5.2).

3 Use Cases

The use cases in this section apply to TAXII Clients (TXC) that connect to a TAXII Server (TXS). For further details on which tests are required for interoperability, refer to section 4: Persona Checklist.

3.1 Authentication and Authorization

TAXII implements Authentication and Authorization as described in section 2.2. The first two tests below verify that the TXC and TXS personas handle authorization parameter errors; the third test verifies they handle certificate-based authentication.

3.1.1 Missing Authorization Parameter Test Case

This test verifies that the TXS will respond with the appropriate error to client requests that are missing the authorization parameter, and that the TXC receives the error message. Table 2 provides an example TXC request and TXS response that uses the Server Discovery Endpoint `/taxii2/`.

Table 2 - Missing Authorization Request and Response

TXC Request

```
GET /taxii2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 401 UNAUTHORIZED
Content-Type: application/taxii+json;version=2.1
WWW-Authenticate: Basic realm="taxii", type=1, title="Login to \"apps\"", Basic
realm="simple"

{
  "title": "Unauthorized",
  "http_status": "401"
}
```

3.1.2 Authorization Parameter Error Test Case

This test verifies that the TXS will respond with the appropriate error to client requests that include an incorrect authorization parameter, and that the TXC receives the error response from the TAXII Server. Table 3 provides an example TXC request and TXS response that uses the Server Discovery Endpoint `/taxii2/`.

Table 3 - Incorrect Authorization Parameter Request and Response

TXC Request

```
GET /taxii2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic eererererere==
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 401 UNAUTHORIZED
Content-Type: application/taxii+json;version=2.1
WWW-Authenticate: Basic realm="taxii", type=1, title="Login to \"apps\"", Basic
realm="simple"

{
  "title": "Unauthorized",
  "http_status": "401"
}
```

3.1.3 Certificate-Based Authentication Test Case

This test verifies that the TXC can authenticate to the server using a certificate, and that the TXS can process the request and deliver the appropriate response. Table 4 provides an example TXC request and TXS response that uses the Server Discovery Endpoint `/taxii2/`.

Table 4 - Certificate-based Authentication Request and Response

TXC Request

```
GET /taxii2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: MIIDBDCCAewCAQAwWTEXMBUGA1UEAwwOb2FzaXMtb3Blbi5vcmcxCzAJBgNVBAYT
AlVTMQswCQYDVQQIDAJWQTEPMA0GA1UEBwwGTWNMZWFuMRMwEQYDVQQKDApPQVNJ
UyBPUEVOMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxcCN04UKBCES
KYbox33bR93+5hWdOsdqQW3nWQ9acmk4pxPmKfwdzCwbir1m6tMF5d4HuLTXvga8
+Acdq0CZ8eSjrhdAxX975oaGEL6A/Y8Q8k/1wx+xjChZsyEFqxmUdcRc8T7VtdVp
pF4Erug3CwWilUfgOecgwB/nH/GgrRUjc9fjpAsvT3lHs0Tr90GQutp/pKOnlC17
yndti4UkBlZAePl3q5PtdOtA2glqA+3hmtq/vm1For1UWYJs0TMhS6iw+fgtJk6X
AZklCPDGzRrbr9UK/SW4HHqstAGuqxh6396g7wtYwHj1C1l6u13XM4iu+Ho0argX
oejSA73wfwIDAQABoGYwZAYJKoZIhvcNAQkOMVcwVTAOBgNVHQ8BAf8EBAMCBaAw
IAYDVR0lAQH/BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMCEGA1UdEQQaMBiBFmNv
bnRhY3RAb2FzaXMtb3Blbi5vcmcwDQYJKoZIhvcNAQELBQADggEBAIajLro4f2Yu
2kMeEw7LGNVu2vmLuYpFkRyQamGHx/+NztzoETGvKodIksH3r1dPGJc1ab9rk9iF
uT99svgZUPrEJZ0D1xccCqb6r+3YFTLhwSBXOE4JvRdEstaXUdrkT9Xe90A6ZjX2
BnJ4X0neL6IYBqaG1yrxTLKvyr+OyxDEkL14ZqyfwjDUwoCyt5+62JpElnOuXNQ2
MNui+EJy8usxIKPPvGwWeJonPzEChnZBs8eBQ2PJmDQjDqsuEveIdrTxCccpH+Dm
WFc/3vvQkByhY/RN0eIZ3Lo9G87EGmTKZAx50yKJeKpR40sYfBG13AoaF/P2mh6T
rYzkG63jqL4=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "title": "TAXII Server Under Test",
  "api_roots": [
    "https://10.1.1.10/api1/",
    "/api2/"
  ]
}
```

3.2 Server Discovery

This Endpoint `/taxii2/` provides general information about a TXS, including the advertised API Roots. It's a common entry point for TXCs into the data and services provided by a TXS. For example, TXCs auto-discovering TXSs via the DNS SRV record will be able to automatically retrieve a discovery response for that server by requesting the `/taxii2/` path on that domain. Please see section 4.1 of the TAXII 2.1 OASIS Standard for further details.

3.2.1 Get Discovery Resource Test Case

This test verifies that the TXC persona can request a Discovery Resource, and that the TXS can process the request and deliver the appropriate response. Support for absolute and relative paths is required for TXC, as seen in the response in Table 4.

Table 5 - Get Discovery Resource

TXC Request

```
GET /taxii2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "title": "TAXII Server Under Test",
  "api_roots": [
    "https://10.1.1.10/api1/",
    "/api2/"
  ]
}
```

3.3 Get API Root Information

This Endpoint `{api-root}/` provides general information about an API Root, which can be used to help users and clients decide whether and how they want to interact with it. Multiple API Roots **MAY** be hosted on a single TAXII Server. Often, an API Root represents a single trust group. See section 4.2 of the TAXII 2.1 OASIS Standard for further details.

3.3.1 Get API Root Resource Test Case

This test verifies that the TXC persona can request an API-Root Resource, and that the TXS can process the request and deliver the appropriate response. Table 5 provides an example TXC request and TXS response. The test case shown in Table 5 builds on the required support for relative paths explained in section 3.2.

Table 6 - Get API Root Request and Response

TXC Request

`GET /api2/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "title": "Sharing Group 2",`
`  "description": "This sharing group shares intelligence",`
`  "versions": [ "application/taxii+json;version=-2.1" ],`
`  "max_content_length": 104857600`
`}`

3.3.2 Incorrect API Root Information Test Case

This test verifies that the TXS can process the request and deliver the appropriate error response, and that the TXC can process the error response when making a request with an incorrect API Root. Table 7 provides an example request and response where the API Root requested (`api3`) does not exist (see section 3.4 for information about getting a Collections Resource).

Table 7 - Incorrect API Root Info Request and Response

TXC Request

`GET /api3/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

```
HTTP/1.1 404 Not Found
Content-Type: application/taxii+json;version=2.1

{
  "title": "Not Found",
  "http_status": "404",
}
```

3.4 Get Collections Test Case

This Endpoint `{api-root}/collections/` provides information about the Collections hosted under this API Root. This provides information about all of the Collections. Most importantly, it provides the Collections' `id` properties, which are used to request objects or manifest entries from a Collection. If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). Please see section 5.1 of the TAXII 2.1 OASIS Standard for further details.

3.4.1 Get Collections Resource Test Case

This test verifies that the TXC persona can request a Collections Resource, and that the TXS can process the request and deliver the appropriate response. Table 8 provides an example TXC request and TXS response.

Table 8 - Get Collections Request and Response

TXC Request

```
GET /api1/collections/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "collections": [
    {
      "id": "2d086da7-4bdc-4f91-900e-d77486753710",
      "title": "Some Collection",
      "can_read": true,
      "can_write": false
    },
    {
      "id": "66666666-4bdc-4f91-900e-d77486753710",
      "title": "Some Other Collection",
      "can_read": true,
      "can_write": false
    }
  ]
}
```

3.5 Get a Collection

This Endpoint `{api-root}/collections/{id}/` provides general information about a Collection, which can be used to help users and clients decide whether and how they want to interact with it. For example, it will tell TXCs what it's called and what permissions they have to it. If a TXC fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). Please see section 5.2 of the TAXII 2.1 OASIS Standard for further details.

3.5.1 Get Collection Resource Test Cases

Four different tests, corresponding to different read/write privileges, are used to verify that the TAXII Server will respond to a TAXII Client request for Collection resources, and that the TAXII Client can process the TAXII Server response.

3.5.1.1 Write-only Collection Resource Test Case

This test verifies that the TXC persona can request a write-only Collection Resource, and that the TXS can process the request and deliver the appropriate response. Table 9 provides an example TXC request and TXS response.

Table 9 - Get Collection Resource (Write-Only) Request and Response

TXC Request

```
GET /api1/collections/1105e147-e4c1-4566-8fb1-1046d181fbf8/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "id": "1105e147-e4c1-4566-8fb1-1046d181fbf8",
  "title": "Collection 1",
  "can_read": false,
  "can_write": true
}
```

3.5.1.2 Read-Write Collection Resource Test Case

This test verifies that the TXC persona can request a read-write Collection Resource, and that the TXS can process the request and deliver the appropriate response. Table 10 provides an example TXC request and TXS response.

Table 10 - Get Collection Resource (Read-Write) Request and Response

TXC Request

```
GET /api1/collections/378e5de7-84a4-45e4-8a34-c02a43d0b657/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "id": "378e5de7-84a4-45e4-8a34-c02a43d0b657",
  "title": "Collection 3",
  "can_read": true,
  "can_write": true
}
```

3.5.1.3 Read-only Collection Resource Test Case

This test verifies that the TXC persona can request a read-only Collection Resource, and that the TXS can process the request and deliver the appropriate response. Table 11 provides an example TXC request and TXS response.

Table 11 - Get Collection Resource (Read-Only) Request and Response

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "id": "253900d3-b9dd-46df-8184-469380fae6d2",
  "title": "Collection 2",
  "can_read": true,
  "can_write": false
}
```

3.5.1.4 No-Read-No-Write Collection Resource Test Case

This test verifies that the TXC persona can request a no-read-no-write Collection Resource, and that the TXS can process the request and deliver the appropriate response. Table 12 provides an example TXC request and TXS response.

Table 12 - Get Collection Resource (No-Read-No-Write) Request and Response

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "id": "253900d3-b9dd-46df-8184-469380fae6d2",
  "title": "Collection 4",
  "can_read": false,
  "can_write": false
}
```

3.5.2 Derived Authorization Errors

Four different tests, corresponding to different read/write privileges, are used to verify that the TXS will respond to a TXC request for Collection resources when the TXC does not have the required permission, and that the TXC can process the TXS response.

3.5.2.1 Read Request for Write-only Collection Test Case

This test verifies that the TXC persona can request to read a write-only Collection Resource, and that the TXS can process the request and deliver an HTTP 403 (Forbidden) error response. Examples of no-read collections are given in section 3.5.1.1 and section 3.5.1.4. Table 13 provides an example TXC request and TXS response.

Table 13 - Read Request for Write-only Collection Request and Response

TXC Request

`GET /api1/collections/1105e147-e4c1-4566-8fb1-1046d181fbf8/objects/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 403 Forbidden`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "title": "Forbidden",`
`  "http_status": "403"`
`}`

3.5.2.2 Write Request to Read-only Collection Test Case

This test verifies that the TXC persona can request to write a read-only Collection Resource, and that the TXS can process the request and deliver an HTTP 403 (Forbidden) error response. Examples of no-write collections are given in section 3.5.1.3 and section 3.5.1.4. Table 14 provides an example TXC request and TXS response.

Table 14 - Write Request to Read-only Collection Request and Response

TXC Request

```
POST /api1/collections/1105e147-e4c1-4566-8fb1-1046d181fbf8/objects/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
Content-Type: application/taxii+json;version=2.1
User-Agent: TAXII-Client/2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--252c7c11-daf2-42bd-843b-be65edca9f61",
      "spec_version": "2.1",
      "name": "Bad IP1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value = '198.51.100.1' ]",
      "pattern_type": "stix"
    }
  ]
}
```

TXS Response

`HTTP/1.1 403 Forbidden`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "title": "Forbidden",`
`  "http_status": "403"`
`}`

3.5.2.3 Delete Request to Read-only or Write-only Collection Test Case

This test verifies that the TXC persona can request to delete a read-only or write-only Collection Resource, and that the TXS can process the request and deliver an HTTP 403 (Forbidden) error response. An example of a write-only collection is given in section 3.5.1.1 and an example of a read-only collection is given in section 3.5.1.3. Table 15 provides an example TXC request and TXS response.

Table 15 - Delete Request to Read-only or Write-only Collection Request and Response

TXC Request

`DELETE /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/`
`indicator--252c7c11-daf2-42bd-843b-be65edca9f61/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 403 Forbidden`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "title": "Forbidden",`
`  "http_status": "403"`
`}`

3.5.2.4 Delete Request to No-Read, No-Write Collection Test Case

This test verifies that the TXC persona can request to delete a no-read, no-write Collection Resource, and that the TXS can process the request and deliver an HTTP 404 (Not Found) error response. An example of a no-read, no-write collection is given in section 3.5.1.4. Table 16 provides an example TXC request and TXS response.

Table 16 - Write Request to Read-Only Collection Request and Response

TXC Request
`DELETE /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/`

```
indicator--252c7c11-daf2-42bd-843b-be65edca9f61/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 404 Not Found
Content-Type: application/taxii+json;version=2.1

{
  "title": "Not Found",
  "http_status": "404"
}
```

3.5.3 Incorrect Collection Information Test Case

This test verifies that the TXC persona can request a Collection Resource using an incorrect Collection ID, and that the TXS can process the request and deliver an HTTP 404 (Not Found) error response, which the TXC can process. Table 17 provides an example TXC request and TXS response where the Collection, */api1/collections/d021ecc8-ab8e-41ab-815e-911c7e329f88/*, does not exist.

Table 17 - Incorrect Collection Info Request and Response

TXC Request

```
GET /api1/collections/d021ecc8-ab8e-41ab-815e-911c7e329f88/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 404 Not Found
Content-Type: application/taxii+json;version=2.1

{
  "title": "Not Found",
  "http_status": "404"
}
```

3.6 Get Object Manifests

This Endpoint `{api-root}/collections/{id}/manifest/` retrieves a manifest about the objects in a Collection. It supports filtering identical to the Get Objects Endpoint, but rather than returning the object itself it returns metadata about the object. It can be used to retrieve metadata to decide whether it's worth retrieving the actual objects.

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). If the Collection specifies `can_read` as `false` for a particular client, this Endpoint **MUST** return an HTTP 403 (Forbidden) error.

See section 5.3 of the TAXII 2.1 OASIS Standard for further details. This endpoint supports filtering; see section 3.13 for details (example given in section 3.13.1.4). This endpoint supports pagination; for details see section 3.14. The common use case requirements from section 2.1 are applicable.

3.6.1 Get Manifest Resource Test Case

This test verifies that the TXC persona can request an Object Manifests Resource, and the TXS can process the request and deliver the appropriate response. Table 18 provides an example TXC request and TXS response.

Table 18 - Get Object Manifests Request and Response

TXC Request

```
GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/manifest/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2016-11-03T12:30:59.000Z
X-TAXII-Date-Added-Last: 2016-11-04T10:29:061Z

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--29aba82c-5393-42a8-9edb-6a2cb1df070b",
      "spec_version": "2.1",
      "name": "Bad IP1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": ["malicious-activity"],
      "pattern": "[ipv4-addr:value = '198.51.100.12']",
      "pattern_type": "stix"
    },
    {
      "type": "indicator",
      "id": "indicator--ef0b28e1-308c-4a30-8770-9b4851b260a5",
      "spec_version": "2.1",
      "name": "Bad IP1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": ["malicious-activity"],
      "pattern": "[ipv4-addr:value = '198.51.100.12']",
      "pattern_type": "stix"
    }
  ]
}
```

3.7 Get Objects

This Endpoint `{api-root}/collections/{id}/objects/` retrieves objects from a Collection. Clients can search for objects in the Collection, retrieve all objects in a Collection, or paginate through objects in the Collection. This is an endpoint for which pagination is applicable; see section 3.14 for details. The common use case requirements from section 2.1 are applicable.

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). If the Collection specifies **`can_read`** as `false` for a particular client, this Endpoint **MUST** return an HTTP 403 (Forbidden) error; an associated test case is given in section 3.5.2.1.

To support searching the Collection, this endpoint supports filtering. Clients can provide one or more filter parameters to get objects with a specific ID, of a specific type, or with a specific version. See section 3.13 for details.

See section 5.4 of the TAXII 2.1 OASIS Standard for further details about this endpoint.

3.7.1 Get Envelope Resource (Get Objects) Test Case

This test verifies that the TXC persona can request all objects from a collection, and the TXS can process the request and deliver the appropriate response.

Two examples are given below: Table 19 provides an example TXC request and TXS response where the TXC can read all three objects in a collection; Table 20 provides an example where the TXC can only read one object.

Table 19 - Get Objects Request and Response (access to all objects)

TXC Request

`GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response
`HTTP/1.1 200 OK`

```
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2018-01-17T11:11:13.000Z
X-TAXII-Date-Added-Last: 2018-01-18T11:11:13.000Z

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--cadd4d85-4ba3-5dd2-9e67-b7bf80bfc471",
      "spec_version": "2.1",
      "name": "Bad IP Subnets",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value ISSUBSET '198.51.100.0/24' OR ipv4-addr:value ISSUBSET
'196.45.200.0/24' ]",
      "pattern_type": "stix"
    },
    {
      "type": "indicator",
      "id": "indicator--57ec1fb8-7a4d-52ef-a18a-4018996dfbba",
      "spec_version": "2.1",
      "name": "Bad IP CIDR",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-18T11:11:13.000Z",
      "modified": "2018-01-18T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value ISSUBSET '198.51.100.0/24' ]",
      "pattern_type": "stix"
    }
  ]
}
```

Table 20 - Get Objects Request and Response (access to one object)

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2018-01-18T11:11:13.000Z
X-TAXII-Date-Added-Last: 2018-01-18T11:11:13.000Z

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--57ec1fb8-7a4d-52ef-a18a-4018996dfbba",
      "spec_version": "2.1",
      "name": "Bad IP CIDR",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-18T11:11:13.000Z",
      "modified": "2018-01-18T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value ISSUBSET '198.51.100.0/24' ]",
      "pattern_type": "stix"
    }
  ]
}
```

3.7.2 No Objects Test Case

This test verifies that the TXC persona can request all objects from an empty collection, and the TXS can process the request and deliver the appropriate response. Table 21 provides an example request and response where the collection requested *a346a557-a132-5233-b20e-3143d20a469c* contains no objects.

Table 21 - No Objects Request and Response

TXC Request

`GET /api1/collections/a346a557-a132-5233-b20e-3143d20a469c/objects/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`}`

3.8 Get an Object

This Endpoint `{api-root}/collections/{id}/objects/{object-id}/` gets an object from a Collection by its **`id`**. It can be thought of as a search where the `match[id]` parameter is set to the `{object-id}` in the path (see section 3.13 for filtering information).

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). If the Collection specifies `can_read` as `false` for a particular client, this Endpoint **MUST** return an HTTP 403

(Forbidden) error. To support getting a particular version of an object, this Endpoint supports filtering as defined in section 3.13.

See section 5.6 of the TAXII 2.1 OASIS Standard for further details. This endpoint supports filtering; see section 3.13 for details. This endpoint supports pagination; for details see section 3.14. The common use case requirements from section 2.1 are applicable.

3.8.1 Get Envelope Resource (Get an Object) Test Case

This test verifies that the TXC persona can request an object from a collection, and the TXS can process the request and deliver the appropriate response. Table 22 provides an example TXC request and TXS response.

Table 22 - Get an Object Request and Response

TXC Request

`GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/`
`indicator--252c7c11-daf2-42bd-843b-be65edca9f61/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`
`X-TAXII-Date-Added-First: 2020-12-03T12:30:59.000Z`
`X-TAXII-Date-Added-Last: 2020-12-03T12:30:59.000Z`
 	
`{`
`  "objects": [`
`    {`
`      "type": "indicator",`
`      "id": "indicator--252c7c11-daf2-42bd-843b-be65edca9f61",`
`      "spec_version": "2.1",`
`      "name": "Bad IPv6-1",`
`      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",`
`      "created": "2020-12-03T12:30:59.000Z",`
`      "modified": "2020-12-03T12:30:59.000Z",`
`      "valid_from": "2020-01-01T00:00:00.000Z",`
`      "indicator_types": [ "malicious-activity" ],`
`      "pattern": "[ ipv6-addr:value = '2001:0db8:85a3:0000:0000:8a2e:0370:7334' ]",`
`      "pattern_type": "stix"`
`    }`
`  ]`
`}`

3.8.2 Object Not Found Test Case

This test verifies that the TXC persona can request a non-existent object from a collection, and the TXS can process the request and deliver the appropriate response. Table 23 provides an example request and response where the object requested *`indicator--252c7c11-daf2-42bd-843b-be65edca9f61`* does not exist.

Table 23 - Object Not Found Request and Response

TXC Request

`GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/`
`indicator--258e7d43-ae46-5081-bd12-bf09ab41b1ee/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 404 Not Found`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "title": "Not Found",`
`  "http_status": "404"`
`}`

3.9 Get Object Versions

This Endpoint `{api-root}/collections/{id}/objects/{object-id}/versions/` retrieves a list of one or more versions of an object in a Collection. This list can be used to decide whether it's worth retrieving the actual objects, or if new versions have been added. If a STIX object is not versioned (and therefore does not have a modified timestamp), the server **MUST** use the **`created`** timestamp. See section 5.8 of the TAXII 2.1 OASIS Standard for further details.

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Not Found) error. And if the Collection specifies **`can_read`** as false for a particular client, this endpoint **MUST** return an HTTP 403 (Forbidden) error.

This endpoint supports filtering; see section 3.13 for details. This endpoint supports pagination; for details see section 3.14. The common use case requirements from section 2.1 are applicable.

3.9.1 Get Versions Resource Test Case

This test verifies that the TXC persona can request a list of one or more versions of an object in a collection, and the TXS can process the request and deliver the appropriate response. Table 24 provides an example TXC request and TXS response.

Table 24 - Get Object Versions

TXC Request

```
GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/
indicator--252c7c11-daf2-42bd-843b-be65edca9f61/versions/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2020-11-03T12:30:59.000Z
X-TAXII-Date-Added-Last: 2020-12-03T12:30:59.000Z

{
  "versions": [
    "2020-11-03T12:30:59.000Z",
    "2020-12-03T12:30:59.000Z"
   ]
}
```

3.10 Add Objects

This Endpoint `{api-root}/collections/{id}/` adds objects to a Collection.

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized). If the Collection specifies `can_write` as `false` for a particular client, this Endpoint **MUST** return an HTTP 403 (Forbidden) error; an associated test case is given in section 3.5.2.2.

Please see section 5.5 of the TAXII 2.1 OASIS Standard for further details.

3.10.1 Add Envelope Resource Test Case

This test verifies that the TXC persona can add objects to a collection, and the TXS can process the request and deliver the appropriate response. The TXS response **MUST** be processed to verify the correct total count of objects (**`total_count`**). In addition, the success count (**`success_count`**) **MUST** equal total count, and failure count (**`failure_count`**) and pending count (**`pending_count`**) **MUST** be zero. Table 25 provides an example TXC request and TXS response[1].

[1] *The UUID shown corresponds to the write-only collection. If the test is being performed for a write-read collection, then replace the UUID with an appropriate collection UUID.*

Table 25 - Indicator Publication POST Request and Response

TXC Request

`POST /api1/collections/1105e147-e4c1-4566-8fb1-1046d181fbf8/objects/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`Content-Type: application/taxii+json;version=2.1`
`User-Agent: TAXII-Client/2.1`
 	
`{`
 	
`  "objects": [`
`    {`
`      "type": "indicator",`
`      "id": "indicator--252c7c11-daf2-42bd-843b-be65edca9f61",`
`      "spec_version": "2.1",`
`      "name": "Bad IP1",`
`      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",`
`      "created": "2018-01-17T11:11:13.000Z",`
`      "modified": "2018-01-17T11:11:13.000Z",`
`      "valid_from": "2018-01-01T00:00:00.000Z",`
`      "indicator_types": [ "malicious-activity" ],`
`      "pattern": "[ ipv4-addr:value = '198.51.100.1' ]",`
`      "pattern_type": "stix"`
`    }`
`  ]`
`}`

TXS Response

`HTTP/1.1 202 Accepted`
`Content-Type: application/taxii+json;version=2.1`
 	
`{`
`  "id": "2d086da7-4bdc-4f91-900e-d77486753710",`
`  "status": "complete",`
`  "total_count": 1,`
`  "success_count": 1,`
`  "failure_count": 0,`
`  "pending_count": 0`
`}`

3.11 Get Status

This Endpoint `{api-root}/status/{status-id}/` provides information about the status of a previous request. In the TAXII 2.1 OASIS Standard, the only request that can be monitored is one to add objects to a Collection. It is typically used by TXCs to monitor a POST request that they made in order to take action when it is complete. TXS **MUST** accept queries for a given status ID for at least 24 hours after the server has finished processing the request. See section 4.3 of the TAXII 2.1 OASIS Standard for further details.

3.11.1 Get Status Resource Test Case

This test verifies that the TXC persona can request the status of a prior request, and the TXS can process the request and deliver the appropriate response. Table 26 provides an example TXC request and TXS response.

Table 26 - Get API Root Status Request and Response

TXC Request

```
GET /api1/status/2d086da7-4bdc-4f91-900e-d77486753710/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "id": "2d086da7-4bdc-4f91-900e-d77486753710",
  "status": "pending",
  "total_count": 4,
  "success_count": 2,
  "failure_count": 1,
  "pending_count": 1
}
```

3.11.2 Get Complete Status Resource Test Case

This test verifies that the TXC persona can request the status of a prior request, and the TXS can process the request and deliver the appropriate response. Table 27 provides an example TXC request and TXS response.

Table 27 - Get API Root Status Request and Response

TXC Request

```
GET /api1/status/2d086da7-4bdc-4f91-900e-d77486753710/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
```

```
{
  "id": "2d086da7-4bdc-4f91-900e-d77486753710",
  "status": "pending",
  "request_timestamp": "2016-11-02T12:34:34.12345Z",
  "total_count": 3,
  "success_count": 1,
  "successes": [
    {
      "id": "indicator--c410e480-e42b-47d1-9476-85307c12bcbf",
      "version": "2022-01-01T12:02:41.312Z"
    }
  ],
  "failure_count": 1,
  "failures": [
    {
      "id": "indicator--19ef5a33-ef0f-43e0-82e6-8fdb02fb1fb0",
      "version": "2022-01-02T12:02:41.312Z"
    }
  ],
  "pending_count": 1,
  "pendings": [
    {
      "id": "indicator--b69a2dbd-6eeb-4a63-8796-80ce4bc2c704",
      "version": "2022-01-01T12:03:41.312Z"
    }
  ]
}
```

3.12 Delete an Object

This Endpoint `{api-root}/collections/{id}/objects/{object-id}/` deletes an object from a Collection by its `id`. Please see section 5.7 of the TAXII 2.1 OASIS Standard for further details.

If a client fails authentication then this endpoint **MUST** return an HTTP 401 (Unauthorized).

If a TXC receives an HTTP 403 error status for this endpoint, then the TXC is recommended to review the **`can_read`** and **`can_write`** permissions it has with the TXS for the particular collection involved. The DELETE endpoint is only supported for collections where both **`can_read`** and **`can_write`** are true.

An HTTP 403 error is returned on this endpoint when only one of **`can_read`** and **`can_write`** is true; an associated test case is given in section 3.5.2.3.

An HTTP 404 error is returned on this endpoint when both **`can_read`** is true and **`can_write`** are false; an associated test case is given in section 3.5.2.4.

3.12.1 Delete Test Case

This test verifies that the TXC persona can delete an object from a collection, and the TXS can process the request and deliver the appropriate response. To confirm that the object was successfully deleted, the client should request the object (see section 3.8) and the server's response should be "404 Not Found" (see section 3.8.2). Table 28 provides an example TXC request and TXS response.

Table 28 - Delete Object Request and Response

TXC Request

```
DELETE /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/
indicator--252c7c11-daf2-42bd-843b-be65edca9f61/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
```

3.13 Filter Results

A TXC can request specific content from a TXS by specifying a set of filters included in the request to the server. Please see section 3.4 of the TAXII 2.1 OASIS Standard for details.

Section 3.13.1 gives test cases for basic filtering as defined in the TAXII specification. The TAXII specification defines four URL query parameters (`add_after`, `limit`, `next`, `match[<field>]`) and four match fields (`id`, `type`, `version`, `spec_version`). Clients **MUST** be able to generate requests with multiple values for a single match parameter, and servers **MUST** be able to handle such requests; associated test cases are also given in section 3.13.1. Test cases for filtering with additional match fields are given in section 3.13.2.

3.13.1 Basic Filtering

Basic URL filtering parameters are not applicable to all Endpoints. The Endpoints to which filtering applies are shown in Table 29.

Table 29 - Endpoint Use of URL Filtering Parameters

URL Filtering Parameter	Get Object Manifests	Get Objects	Get an Object	Get Object Versions	Delete an Object
`added_after`	X	X	X	X	
`limit`	X	X	X	X	
`next`	X	X	X	X	
`match[id]`	X	X			

`match[type]`	X	X			
`match[version]`	X	X	X		X
`match[spec_version]`	X	X	X	X	X

An example for each of the URL filtering parameters is given below. While examples are not given for all Endpoints, the format and use is similar. Notes specific to each Endpoint type are as follows:

- The Get Object Manifests Endpoint supports the same filters as the Get Objects Endpoint. Filtering is applied against the source object rather than the manifest entry for an object. Thus, searching the manifest where **type** equals `indicator` will return the manifest entries for *objects* of `indicator` type, even though a manifest doesn't have a **type** property.
- The Get Objects Endpoint supports filtering a Collection. TAXII Clients can provide one or more filter parameters to get objects with a specific ID, of a specific type, or with a specific version.
- The Get an Object Endpoint uses `match[version]` to retrieve a particular version of an object.
- The Delete an Object Endpoint uses `match[version]` to support removing a particular version of an object.
- The `added_after` parameter is in no way related to dates or times in a STIX object or any other CTI object.
- The Get an Object Endpoint is equivalent to filtering the Get Objects Endpoint where the `match[id]` parameter is set to the **`{object-id}`** in the path.

3.13.1.1 added_after Test Case

This test verifies that the TXC persona can request objects that were added after a specified timestamp and the TXS can process the request and deliver the appropriate response by filtering the results. Table 30 provides an example TXC request and TXS response that includes the `added_after` URL query parameter.

Table 30 - Get Objects Request and Response with added_after URL Query Parameter

TXC Request

`GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?added_after=2021-11-05T 10:30:061Z HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`
`X-TAXII-Date-Added-First: 2019-01-17T11:11:13.000Z`

```
X-TAXII-Date-Added-Last: 2019-01-17T11:11:13.000Z

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--11dabf1d-71a8-42f4-aa97-2c8a5962f697",
      "spec_version": "2.1",
      "name": "Malicious URL",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2019-01-17T11:11:13.000Z",
      "modified": "2019-01-17T11:11:13.000Z",
      "valid_from": "2019-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ url:value = 'https://www.9a6.info/bar' ]",
      "pattern_type": "stix"
    }
  ]
}
```

When using this filter with the Get an Object Endpoint, the object requested will only be returned if it was added after the specified timestamp. Table 31 shows the response when the object was added before the specified timestamp (no object returned).

Table 31 - Get an Object Request and Response with added_after URL Query Parameter

TXC Request

`GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/ indicator--252c7c11-daf2-42bd-843b-be65edca9f61/?added_after=2021-11-05T10:30:061Z HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`

3.13.1.2 limit Test Case

This test verifies that the TXC persona can limit the number of objects returned in a request, and the TXS can process the request and deliver the appropriate response by filtering the results. For brevity, the process of pagination to obtain the remaining results is not shown in this test case; an example of the pagination process is given in section 3.14. Table 32 provides an example TXC request and TXS response that includes the `limit` URL query parameter.

Table 32 - Get Object Manifests Request and Response: limit URL Query Parameter

TXC Request

```
GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/manifest/?limit=2 HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2018-01-17T11:11:13.000Z
X-TAXII-Date-Added-Last: 2018-01-19T11:11:13.000Z

{
  "more": true,
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--69a4eedb-05c5-463b-ba59-65257d652cf4",
      "spec_version": "2.1",
      "name": "Bad Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ domain-name:value = 'www.5z8.info' ]",
      "pattern_type": "stix"
    },
    {
      "type": "indicator",
      "id": "indicator--7d663616-ab3d-4097-b195-ace869edefc5",
      "spec_version": "2.1",
      "name": "Not Good Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-19T11:11:13.000Z",
      "modified": "2018-01-19T11:11:13.000Z",
      "valid_from": "2018-01-21T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ domain-name:value = 'www.2z9.info' ]",
      "pattern_type": "stix"
    }
  ]
}
```

3.13.1.3 match[id] Test Case

This test verifies that the TXC persona can request objects that match a given identifier (`id`), and the TXS can process the request and deliver the appropriate response by filtering the results. Table 33 provides an example TXC request and TXS response.

Table 33 - Get Objects Request and Response with match[id]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[id]=indicator
--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2018-03-17T11:11:13.000Z
X-TAXII-Date-Added-Last: 2018-03-17T11:11:13.000Z

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--fb07bb1e-9745-489f-9a4c-b17bf1e7aab1",
      "spec_version": "2.1",
      "name": "Possibly Malicious Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-03-17T11:11:13.000Z",
      "modified": "2018-03-17T11:11:13.000Z",
      "valid_from": "2018-02-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ domain-name:value = 'www.1234foobar.info' ]",
      "pattern_type": "stix"
    }
  ]
}
```

3.13.1.4 match[type] Test Cases

Two test cases are given below for filtering on `type`. The first filters a Get Objects request; the second filters a Get Objects Manifest request.

This test verifies that the TXC persona can request objects that match a given type (`type`), and the TXS can process the request and deliver the appropriate response by filtering the results. Table 34 provides an example TXC request and TXS response.

Table 34 - Get Objects Request and Response with match[type]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[type]=indicator
HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--6cce5ca8-34c0-4ae8-b603-0bda82504dfd",
      "spec_version": "2.1",
      "name": "Bad URL or Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ url:value = 'https://www.5z8.info/foo' OR domain-name:value =
'www.5z8.info' ]",
      "pattern_type": "stix"
    },
    {
      "type": "campaign",
      "id": "campaign--76a9f73c-c61b-4079-8cef-7a6246238b4e",
      "spec_version": "2.1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-06T20:03:00.000Z",
      "modified": "2016-04-06T20:03:00.000Z",
      "name": "Green Group Attacks Against Finance"
    }
  ]
}
```

This test verifies that the TXC persona can request objects that match a given type (`type`), and the TXS can process the request and deliver the appropriate response by filtering the results. Table 35 provides an example TXC request and TXS response.

Table 35 - Get Object Manifest Request and Response with match[type]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/manifest/?match[type]=indicator
HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response
(see below)

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--dd547b86-2880-41c9-a8db-0fa4f4977bae",
      "spec_version": "2.1",
      "name": "Bad URL or Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2019-01-17T11:11:13.000Z",
      "modified": "2019-01-17T11:11:13.000Z",
      "valid_from": "2019-01-01T00:00:00.000Z",
      "indicator_types": ["malicious-activity"],
      "pattern": "[url:value = 'https://www.1a2.info/bar']",
      "pattern_type": "stix"
    },
    {
      "type": "campaign",
      "id": "campaign--245ca48e-c114-4f38-999e-e65c70a4c371",
      "spec_version": "2.1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2017-04-06T20:03:00.000Z",
      "modified": "2017-04-06T20:03:00.000Z",
      "name": "Red Group Attacks Against Manufacturing"
    }
  ]
}
```

3.13.1.5 match[version] Test Case

This test verifies that the TXC persona can request objects that match a given version (`version`), and the TXS can process the request and deliver the appropriate response by filtering the results. Table 36 provides an example TXC request and TXS response.

The version is determined by an object's **modified** timestamp. If an object is not versioned and therefore does not have a **modified** date property, then the version **MUST** be determined by the **created** timestamp. If an object does not have a **created** or **modified** timestamp, then the version **MUST** be determined by the **date_added** timestamp of the Manifest-Record resource (see section 5.3.1 of the TAXII 2.1 OASIS Standard), which is when the object was added to the server.

Table 36 - Get Objects Request and Response with match[version]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[version]=last
HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--cca8e422-8f15-48de-98f7-d0ab7767acfe",
      "spec_version": "2.1",
      "name": "Bad URL or Domain",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2019-02-17T11:11:13.000Z",
      "modified": "2019-02-17T11:11:13.000Z",
      "valid_from": "2019-02-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ url:value = 'https://www.2a1.info/foobar' ]",
      "pattern_type": "stix"
    },
    {
      "type": "campaign",
      "id": "campaign--f4aaed32-79b7-455b-8ef7-a79ca33f5d7d",
      "spec_version": "2.1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2017-05-06T20:03:00.000Z",
      "modified": "2017-05-06T20:03:00.000Z",
      "name": "Purple Group Attacks Against Retail"
    }
  ]
}
```

3.13.1.6 match[spec_version] Test Case

This test verifies that the TXC persona can delete objects that match a given specification version (`spec_version`), and the TXS can process the request and deliver the appropriate response by filtering the results. Table 37 provides an example TXC request and TXS response that includes the `spec_version` URL query parameter.

Table 37 - Delete Object Request and Response

TXC Request

```
DELETE /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/
indicator--252c7c11-daf2-42bd-843b-be65edca9f61/?match[spec_version]=2.1 HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
```

`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`

3.13.1.7 Logical OR Operator Test Case

This test verifies that the TXC persona can utilize the logical OR operator in a request, and the TXS can process the request and deliver the appropriate response by filtering the results. Table 38 provides an example TXC request and TXS response.

Table 38 - Get Objects Request and Response using logical OR operator

TXC Request

`GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[type]=campaign,`
`threat-actor HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "campaign",
      "spec_version": "2.1",
      "id": "campaign--208f342f-1f18-48a1-a898-0a66b04f1b7d",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2020-04-06T20:03:00.000Z",
      "modified": "2020-04-06T20:03:00.000Z",
      "name": "Brown Group Attacks Against Commerce"
    },
    {
      "type": "campaign",
      "spec_version": "2.1",
      "id": "campaign--a4b9e39d-f51f-4c47-8cda-61852cbf93d3",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2021-04-06T20:03:00.000Z",
      "modified": "2021-04-06T20:03:00.000Z",
      "name": "Pink Group Attacks Against Energy"
    },
    {
      "type": "threat-actor",
      "spec_version": "2.1",
      "id": "threat-actor--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-06T20:03:48.000Z",
      "modified": "2016-04-06T20:03:48.000Z",
      "threat_actor_types": [ "crime-syndicate" ],
      "name": "Evil Org",
      "description": "The Evil Org threat actor group",
      "roles": [ "director" ],
      "sophistication": "advanced",
      "resource_level": "team",
      "primary_motivation": "organizational-gain"
    }
  ]
}
```

3.13.1.8 Logical AND Operator Test Case

This test verifies that the TXC persona can utilize the logical AND operator in a request, and the TXS can process the request and deliver the appropriate response by filtering the results. Table 39 provides an example TXC request and TXS response.

Table 39 - Get Objects Request and Response using logical AND operator

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[type]=incident&
match[version]=2021-01-03T01:01:01.000Z HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--e9af88c8-e101-413a-a8d1-f869ad6d79b3",
      "spec_version": "2.1",
      "name": "Bad IP2",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2022-01-17T11:11:13.000Z",
      "modified": "2022-01-17T11:11:13.000Z",
      "valid_from": "2022-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value = '199.55.102.3' ]",
      "pattern_type": "stix"
    }
  ]
}
```

3.13.1.9 Logical OR and AND Operators Test Case

This test verifies that the TXC persona can utilize the logical OR and AND operators in a request, and the TXS can process the request and deliver the appropriate response by filtering the results. Table 40 provides an example TXC request and TXS response.

Table 40 - Get Objects Request and Response using logical OR and AND operators

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[type]=campaign,
malware&match[version]=first,last HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "malware",
      "spec_version": "2.1",
      "id": "malware--6f8a1ea6-6655-492b-a5e1-8d02b993b10e",
      "created": "2019-05-12T08:17:27.000Z",
      "modified": "2019-05-12T08:17:27.000Z",
      "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
      "name": "Cryptolocker",
      "malware_types": [ "ransomware" ],
      "is_family": false,
      "capabilities": [ "anti-vm" ],
      "first_seen": "2021-01-18T11:11:13.000Z",
      "last_seen": "2021-01-18T11:11:13.000Z",
      "implementation_langauges": [ "python", "c" ],
      "architecture_execution_envs": [ "mips", "x86" ]
    },
    {
      "type": "campaign",
      "spec_version": "2.1",
      "id": "campaign--ff24310c-7dd9-4768-826d-b6fcd6519cdb",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2021-07-06T20:03:00.000Z",
      "modified": "2021-07-06T20:03:00.000Z",
      "name": "Magenta Group Attacks Against Finance"
    }
  ]
}
```

3.13.2 Filtering with Additional Match Fields

Additional match fields can be used with the Get Object Manifests and Get Objects Endpoints. Three classes of additional match fields are defined (see Appendix B: TAXII Additional Match Filters). The three tiers in the Tiered class **MUST** be verified in sequence. Test cases for each class are given below.

3.13.2.1 Tier 1 Test Case

This test verifies that the TXC persona can request objects using a Tier 1 match field (e.g., `confidence`) and the TXS can process the request and deliver the appropriate response by filtering the results. Table 41 provides an example TXC request and TXS response. See the Tier 1 section of Appendix B for all Tier 1 filters.

Table 41 - Get Objects Request and Response with match[confidence]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[confidence]=90,91,
92,93,94 HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--844cf084-2229-45d5-9764-c3c6ed978d77",
      "spec_version": "2.1",
      "confidence": 90,
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2020-02-06T20:03:48.000Z",
      "modified": "2020-02-06T20:03:48.000Z",
      "indicator_types": [ "benign" ],
      "name": "Benign site",
      "pattern": "[ url:value = 'http://weibo.com' ]",
      "pattern_type": "stix",
      "valid_from": "2020-01-01T00:00:00Z"
    },
    {
      "type": "campaign",
      "id": "campaign--5bd6a633-769b-4211-8f6d-a4567941e4c1",
      "spec_version": "2.1",
      "confidence": 93,
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2020-04-06T20:03:00.000Z",
      "modified": "2020-04-06T20:03:00.000Z",
      "name": "Yellow Group Attacks Against Finance"
    }
  ]
}
```

3.13.2.2 Tier 2 Test Case

This test verifies that the TXC persona can request objects using a Tier 2 match field (e.g., `capabilities`) and the TXS can process the request and deliver the appropriate response by filtering the results. Table 42 provides an example TXC request and TXS response. See the Tier 2 section of Appendix B for all Tier 2 filters.

Table 42 - Get Objects Request and Response with match[capabilities]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[capabilities]=
emails-spam HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "malware",
      "id": "malware--afae2bf9-c5e3-49d8-8e12-8d4c5829f35f",
      "spec_version": "2.1",
      "capabilities": [ "emails-spam" ],
      "created": "2019-05-12T08:17:27.000Z",
      "modified": "2019-05-12T08:17:27.000Z",
      "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
      "name": "EmailLocker",
      "malware_types": [ "ransomware" ],
      "is_family": false,
      "first_seen": "2017-01-18T11:11:13.000Z",
      "last_seen": "2017-01-18T11:11:13.000Z",
      "implementation_langauges": [ "python", "c" ],
      "architecture_execution_envs": [ "mips", "x86" ]
    },
    {
      "type": "malware",
      "id": "malware--f2e6e92c-2979-49d6-b52e-7a07d2bd38b4",
      "spec_version": "2.1",
      "capabilities": [ "emails-spam" ],
      "created": "2019-05-12T08:17:27.000Z",
      "modified": "2019-05-12T08:17:27.000Z",
      "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
      "name": "KeyLogger",
      "malware_types": [ "keylogger" ],
      "is_family": false,
      "first_seen": "2017-01-18T11:11:13.000Z",
      "last_seen": "2017-01-18T11:11:13.000Z",
      "implementation_langauges": [ "python", "c" ],
      "architecture_execution_envs": [ "mips", "x86" ]
    }
  ]
}
```

3.13.2.3 Tier 3 Test Case

This test verifies that the TXC persona can request objects using a Tier 3 match field (e.g., `service_status`) and the TXS can process the request and deliver the appropriate response by filtering the results. Table 43 provides an example TXC request and TXS response. See the Tier 3 section of Appendix B for all Tier 3 filters.

Table 43 - Get Objects Request and Response with match[service_status]

TXC Request

`GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[service_status]= SERVICE_STOPPED HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "process",
      "spec_version": "2.1",
      "id": "process--70b17c6c-93e5-4c80-8683-5a4d4e51f2c1",
      "pid": 2217,
      "command_line": "C:\\Windows\\System32\\sirvizio.exe /s",
      "image_ref": "file--3916128d-69af-5525-be7a-99fac2383a59",
      "extensions": {
        "windows-service-ext": {
          "service_name": "sirvizio",
          "display_name": "Sirvizio",
          "start_type": "SERVICE_AUTO_START",
          "service_type": "SERVICE_WIN32_OWN_PROCESS",
          "service_status": "SERVICE_RUNNING"
        }
      }
    }
  ]
}
```

3.13.2.4 Relationships Test Case

This test verifies that the TXC persona can request objects using a Relationships match field (e.g., `relationships-all`) and the TXS can process the request and deliver the appropriate response by filtering the results. Table 44 provides an example TXC request and TXS response. See the Relationships Match section of Appendix B for all Relationships Match filters.

Table 44 - Get Objects Request and Response with match[relationships-all]

TXC Request

```
GET
/api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[relationships-all]=
indicator--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "sighting",
      "spec_version": "2.1",
      "id": "sighting--ee20065d-2555-424f-ad9e-0f8428623c75",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-06T20:08:31.000Z",
      "modified": "2016-04-06T20:08:31.000Z",
      "sighting_of_ref": "indicator--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb",
      "count": 50,
      "first_seen": "2017-12-21T19:00:00.000Z",
      "last_seen": "2018-01-06T19:00:00.000Z",
    },
    {
      "type": "relationship",
      "spec_version": "2.1",
      "id": "relationship--44298a74-ba52-4f0c-87a3-1824e67d7fad",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-06T20:06:37.000Z",
      "modified": "2016-04-06T20:06:37.000Z",
      "relationship_type": "indicates",
      "source_ref": "indicator--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb",
      "target_ref": "malware--31b940d4-6f7f-459a-80ea-9c1f17b5891b"
    }
  ]
}
```

3.13.2.5 Calculation Test Case

This test verifies that the TXC persona can request objects using a Calculation match field (e.g., `confidence-gte`) and the TXS can process the request and deliver the appropriate response by filtering the results. Table 45 provides an example TXC request and TXS response. See the Calculation Match section of Appendix B for all Calculation Match filters.

Table 45 - Get Objects Request and Response with match[confidence-gte]

TXC Request

```
GET /api1/collections/253900d3-b9dd-46df-8184-469380fae6d2/objects/?match[confidence-gte]=
90 HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
User-Agent: TAXII-Client/2.1
```

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb",
      "spec_version": "2.1",
      "confidence": 90,
      "name": "Bad URL",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ url:value = 'https://www.3a1.info/foobar' ]",
      "pattern_type": "stix"
    },
    {
      "type": "campaign",
      "id": "campaign--4779a4c3-6f0a-4ec2-90ba-5ee32bde736a",
      "spec_version": "2.1",
      "confidence": 93,
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-06T20:03:00.000Z",
      "modified": "2016-04-06T20:03:00.000Z",
      "name": "Orange Group Attacks Against Healthcare"
    }
  ]
}
```

3.14 Pagination

TAXII 2.1 supports pagination of large result sets on certain endpoints. Pagination is used when a TXS has more content to send to a TXC than will fit in a single TAXII container resource (see section 1.1). This **MAY** be a result of a TXS limitation and/or a TXC-specified limit. Should a TXS have more content than will fit in a single TAXII container resource, the TXS is to use pagination by divvying up the results and sending the content via multiple TAXII container resources. These endpoints return results sorted in ascending order by the date they were added to the collection.

For TXS responses containing a TAXII container resource object that has the **more** property set to true, a timestamp-based approach can be utilized by a TXC to paginate through the remaining results.

Specifically, from the TXS response, a TXC can pass the date/time value from the X-TAXII-Date-Added-Last header, along with the same original query options, as the added_after URL parameter. The value of the header will change with subsequent requests. If a TXS has more results than can fit in a single TAXII container resource, it **MUST** set the value of **more** to true; when there are no remaining records to be requested, the value of **more MUST** be false. A TXC SHOULD NOT provide a value for **more**.

Pagination is applicable for the following endpoints:

URL	Methods	Resource Type
{api-root}/collections/**{id}**/manifest/	GET	`manifest`
{api-root}/collections/**{id}**/objects/	GET	`envelope`
{api-root}/collections/**{id}**/objects/**{object-id}**/	GET	`envelope`
{api-root}/collections/**{id}**/objects/**{object-id}**/versions/	GET	`versions`

For further details, please see section 3.5 of the TAXII 2.1 OASIS Standard.

3.14.1 Get Versions Resource Pagination Test Case

This test case is a follow-on to the test case shown in section 3.9.1; this test case illustrates the process used to paginate through results when a TXC requests to retrieve all versions of a particular object within a collection.

From a TXS perspective, this test case will demonstrate the initial and subsequent TXC requests, and the delivery of all of the results across multiple Versions Resources. For this test case, the TXS has a limit of three versions per Versions Resource. This general process is also used when a TXS responds with Manifest Resources or TAXII Envelopes.

First, the TXC requests all of the versions for a particular object within a particular collection; this object will have five versions. The TXS responds with a single object version and sets **more** to true.

Table 46 - Get Object Versions Initial Request

TXC Request

`GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/indicator--252c7c11-daf2-42bd-843b-be65edca9f61/versions/ HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

```
HTTP/1.1 200 OK
Content-Type: application/taxii+json;version=2.1
X-TAXII-Date-Added-First: 2020-11-03T12:30:59.000Z
X-TAXII-Date-Added-Last: 2020-11-03T12:30:59.000Z

{
  "more": true,
  "versions": [
    "2020-04-03T12:30:59.000Z",
    "2020-05-03T12:30:59.000Z",
    "2020-06-03T12:30:59.000Z"
  ]
}
```

Next, the TXC receives a Versions Resource with **more** set to true, and so the TXC makes another request to obtain the remaining records. This time, the TXC passes the provided value of X-TAXII-Date-Added-Last as the added-after URL parameter.

Table 47 - Get Object Versions Subsequent Request

TXC Request

`GET /api1/collections/91a7b528-80eb-42ed-a74d-c6fbd5a26116/objects/indicator--252c7c11-daf2-42bd-843b-be65edca9f61/versions/?added-after=2020-11-03T12:30:59.000Z HTTP/1.1`
`Host: 10.1.1.10`
`Accept: application/taxii+json;version=2.1`
`Authorization: Basic dGVzdDpQYXNzdzByZCE=`
`User-Agent: TAXII-Client/2.1`

TXS Response

`HTTP/1.1 200 OK`
`Content-Type: application/taxii+json;version=2.1`
`X-TAXII-Date-Added-First: 2020-12-04T12:30:59.000Z`
`X-TAXII-Date-Added-Last: 2020-12-04T12:30:59.000Z`
 	
`{`
`  "versions": [`
`    "2020-11-04T12:30:59.000Z",`
`    "2020-12-04T12:30:59.000Z"`
`  ]`
`}`

3.15 Custom Properties

Custom property names **MUST** start with “x_” followed by a source unique identifier, an underscore, and then the name. For the purposes of Interoperability, the source unique identifier is to be a globally-unique identifier (GUID). The GUID **MUST** be a UUIDv4. The UUID **MUST** be generated according to [RFC 4122].

A TXS that receives a TAXII resource with one or more custom properties it does not understand **MUST** ignore the non-understood properties and continue processing the message. In addition, a TXS **MUST** store the complete responses, including the non-understood properties. Logging policies (e.g., retention, retrieval) are beyond the scope of this document.

A TXC that receives a TAXII resource with one or more custom properties it does not understand **MAY** ignore the non-understood properties, but **MUST** continue processing the message. In addition, a TXC **MUST** store the complete responses, including the non-understood properties. Logging policies (e.g., retention, retrieval) are beyond the scope of this document.

3.15.1 Custom Properties Test Case

The TXC submits a POST request to add an object to a Collection on the TXS. The TXC also includes a custom property named *x_18467e42_04f4_4505_93c8_9f1cf29e1045_test_client*, where "18467e42_04f4_4505_93c8_9f1cf29e1045" is the TXC's GUID. This property is received by but not understood by the TXS, and thus the TXS ignores this property but continues with the remainder of the request.

In a similar fashion, the TXS then responds with content that includes a custom property named *x_f18dd923-7fdd-4c5c-94f3-807f556bce6b_test_server*, where "f18dd923-7fdd-4c5c-94f3-807f556bce6b" is the TXS's GUID. This property is received by but not understood by the TXC, and thus the TXC ignores this property but continues with the remainder of the request. Both the TXS and the TXC log the entirety of the custom property content to internal storage.

Both the server and client receive content with properties they do not understand and yet they're interoperable.

Table 48 - Indicator Publication POST Request and Response

TXC Request

```
POST /api1/collections/1105e147-e4c1-4566-8fb1-1046d181fbf8/objects/ HTTP/1.1
Host: 10.1.1.10
Accept: application/taxii+json;version=2.1
Authorization: Basic dGVzdDpQYXNzdzByZCE=
Content-Type: application/taxii+json;version=2.1
User-Agent: TAXII-Client/2.1

{
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--252c7c11-daf2-42bd-843b-be65edca9f61",
      "spec_version": "2.1",
      "name": "Bad IP1",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2018-01-17T11:11:13.000Z",
      "modified": "2018-01-17T11:11:13.000Z",
      "valid_from": "2018-01-01T00:00:00.000Z",
      "indicator_types": [ "malicious-activity" ],
      "pattern": "[ ipv4-addr:value = '198.51.100.1' ]",
      "pattern_type": "stix"
    }
  ],
  "x_18467e42_04f4_4505_93c8_9f1cf29e1045_test_client": "The Client sends the Server a
custom property."
}
```

TXS Response

```
HTTP/1.1 202 Accepted
Content-Type: application/taxii+json;version=2.1

{
  "id": "2d086da7-4bdc-4f91-900e-d77486753710",
  "status": "complete",
  "total_count": 1,
  "success_count": 1,
  "failure_count": 0,
  "pending_count": 0,
  "x_f18dd923_7fdd_4c5c_94f3_807f556bce6b_test_server": "The Server sends the Client a
custom property."
}
```

4 Persona Checklist

The following checklists summarize all tests that a persona **MUST** conform to within that persona.

4.1 TAXII Client (TXC)

For the purpose of this document, a `TXC` is a software package that connects to a TAXII Server and supports the exchange of CTI.

Any instance being qualified as a `TXC` **MUST** confirm test results for the following use cases.

Table 49 - TAXII Client (TXC) Test Verification List

Test Case	Section	Verification	Results
Missing Authorization Parameter	3.1.1	Mandatory	<fill in>
Authorization Parameter Error	3.1.2	Mandatory	<fill in>
Certificate-Based Authentication	3.1.3	Mandatory	<fill in>
Get Discovery Resource	3.2.1	Mandatory	<fill in>
Get API Root Resource	3.3.1	Mandatory	<fill in>
Incorrect API Root Information	3.3.2	Mandatory	<fill in>
Get Collections Resource	3.4.1	Mandatory	<fill in>
Write-Only Collection Resource	3.5.1.1	Mandatory	<fill in>
Read-Write Collection Resource	3.5.1.2	Mandatory	<fill in>
Read-Only Collection Resource	3.5.1.3	Mandatory	<fill in>
No-Read-No-Write Collection Resource	3.5.1.4	Mandatory	<fill in>
Read Request for Write-only Collection	3.5.2.1	Mandatory	<fill in>
Write Request to Read-only Collection	3.5.2.2	Mandatory	<fill in>
Delete Request to Read-only or Write-only Collection	3.5.2.3	Mandatory	<fill in>
Delete Request to No-Read, No-Write Collection	3.5.2.4	Mandatory	<fill in>
Incorrect Collection Information	3.5.3	Mandatory	<fill in>
Get Manifest Resource	3.6.1	Mandatory	<fill in>
Get Envelope Resource (Get Objects)	3.7.1	Mandatory	<fill in>
No Objects	3.7.2	Mandatory	<fill in>
Get Envelope Resource (Get an Object)	3.8.1	Mandatory	<fill in>
Object Not Found	3.8.2	Mandatory	<fill in>

Get Versions Resource	3.9.1	Mandatory	<fill in>
Add Envelope Resource	3.10.1	Mandatory	<fill in>
Get Status Resource	3.11.1	Mandatory	<fill in>
Get Complete Status Resource	3.11.2	Mandatory	<fill in>
Delete	3.12.1	Mandatory	<fill in>
added_after	3.13.1.1	Mandatory	<fill in>
limit	3.13.1.2	Mandatory	<fill in>
match[id]	3.13.1.3	Mandatory	<fill in>
match[type]	3.13.1.4	Mandatory	<fill in>
match[version]	3.13.1.5	Mandatory	<fill in>
match[spec_version]	3.13.1.6	Mandatory	<fill in>
Logical OR Operator	3.13.1.7	Mandatory	<fill in>
Logical AND Operator	3.13.1.8	Mandatory	<fill in>
Logical OR and AND Operators	3.13.1.9	Mandatory	<fill in>
Tier 1	3.13.2.1	Mandatory	<fill in>
Tier 2	3.13.2.2	Mandatory	<fill in>
Tier 3	3.13.2.3	Mandatory	<fill in>
Relationships	3.13.2.4	Mandatory	<fill in>
Calculation	3.13.2.5	Mandatory	<fill in>
Get Versions Resource Pagination	3.14.1	Mandatory	<fill in>
Custom Properties	3.15.1	Mandatory	<fill in>

4.2 TAXII Server (TXS)

For the purpose of this document, a `TXS` is a software package that supports the exchange of CTI.

Any instance being qualified as a `TXS` **MUST** confirm test results for the following use cases.

Table 50 — TAXII Server (TXS) Test Verification List

Test Case	Section	Verification	Results
Missing Authorization Parameter	3.1.1	Mandatory	<fill in>
Authorization Parameter Error	3.1.2	Mandatory	<fill in>
Certificate-Based Authentication	3.1.3	Optional	<fill in>
Get Discovery Resource	3.2.1	Mandatory	<fill in>
Get API Root Resource	3.3.1	Mandatory	<fill in>
Incorrect API Root Information	3.3.2	Mandatory	<fill in>
Get Collections Resource	3.4.1	Mandatory	<fill in>
Write-Only Collection Resource	3.5.1.1	Mandatory	<fill in>
Read-Write Collection Resource	3.5.1.2	Mandatory	<fill in>
Read-Only Collection Resource	3.5.1.3	Mandatory	<fill in>
No-Read-No-Write Collection Resource	3.5.1.4	Mandatory	<fill in>
Read Request for Write-only Collection	3.5.2.1	Mandatory	<fill in>
Write Request to Read-only Collection	3.5.2.2	Mandatory	<fill in>
Delete Request to Read-only or Write-only Collection	3.5.2.3	Mandatory	<fill in>
Delete Request to No-Read, No-Write Collection	3.5.2.4	Mandatory	<fill in>
Incorrect Collection Information	3.5.3	Mandatory	<fill in>
Get Manifest Resource	3.6.1	Mandatory	<fill in>
Get Envelope Resource (Get Objects)	3.7.1	Mandatory	<fill in>
No Objects	3.7.2	Mandatory	<fill in>
Get Envelope Resource (Get an Object)	3.8.1	Mandatory	<fill in>
Object Not Found	3.8.2	Mandatory	<fill in>

Get Versions Resource	3.9.1	Mandatory	<fill in>
Add Envelope Resource	3.10.1	Mandatory	<fill in>
Get Status Resource	3.11.1	Mandatory	<fill in>
Get Complete Status Resource	3.11.2	Optional	<fill in>
Delete	3.12.1	Mandatory	<fill in>
added_after	3.13.1.1	Mandatory	<fill in>
limit	3.13.1.2	Mandatory	<fill in>
match[id]	3.13.1.3	Mandatory	<fill in>
match[type]	3.13.1.4	Mandatory	<fill in>
match[version]	3.13.1.5	Mandatory	<fill in>
match[spec_version]	3.13.1.6	Mandatory	<fill in>
Logical OR Operator	3.13.1.7	Mandatory	<fill in>
Logical AND Operator	3.13.1.8	Mandatory	<fill in>
Logical OR and AND Operators	3.13.1.9	Mandatory	<fill in>
Tier 1	3.13.2.1	Mandatory	<fill in>
Tier 2	3.13.2.2	Mandatory	<fill in>
Tier 3	3.13.2.3	Mandatory	<fill in>
Relationships	3.13.2.4	Mandatory	<fill in>
Calculation	3.13.2.5	Mandatory	<fill in>
Get Versions Resource Pagination	3.14.1	Mandatory	<fill in>
Custom Properties	3.15.1	Mandatory	<fill in>

Appendix A. References

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or version number) and Informative references are either specific or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies. While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long term validity.

Informative References

The following referenced documents are not required for the application of this document but **MAY** assist the user with regard to a particular subject area.

[RFC3986]
Uniform Resource Identifier (URI): Generic Syntax, January 2005, https://www.rfc-editor.org/info/rfc3986.

[RFC4122]
A Universally Unique IDentifier (UUID) URN Namespace, July 2005, https://www.rfc-editor.org/info/rfc4122.

[RFC7540]
Hypertext Transfer Protocol Version 2 (HTTP/2), May 2015, https://www.rfc-editor.org/info/rfc7540.

Appendix B. TAXII Additional Match Fields

Introduction

A TAXII Client can request specific content from a TAXII Server by specifying a set of filters included in the request to the server. Please see the TAXII specification for details [SPEC].

This document focuses on the match URL query parameter, which defines filtering on a specified field. Four match fields are defined in the TAXII specification (`id`, `spec_version`, `type`, `version`). Requests **MAY** use a field not defined in [SPEC], and servers **MAY** ignore fields they do not understand.

This document defines additional fields for the match URL query parameter. Please consider the following when using additional match fields.

- **Special characters**: Any special characters such as white space, question marks, and commas **MUST** be encoded as a character triplet, consisting of the percent character "%" followed by the two hexadecimal digits representing that octet's numeric value [RFC3986].

- **Default values**: Some properties are optional, have default values, and **MAY** not be present. They have specific interpretation in the STIX specification. For example, the **`revoked`** property is optional and if not present, the object is considered valid. The filter `?match[revoked]=false` will return objects that have not been revoked (the **`revoked`** property is not present or equals `false`).

- **List type**: Properties of type `list` can be checked for specific values. If any one of the values in the match filter is present, the object will be returned. For example, consider the `object_refs` field of type `list` of type `identifier`.

```
"object_refs": [
  "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
  "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
  "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a",
  "file--0203b5c8-f8b6-4ddb-9ad0-527d727f968b"
]
```

The filter, `?match[object_refs]=campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c` will return the associated object.

It is not possible to filter for objects that contain a list type field with *all* values in a set because a field **MUST NOT** occur more than once in a filter request.

- **Dictionary type**: Properties of type dictionary can be filtered for specific dictionary key values. For example, consider an X.509 Certificate object with a `hashes` field of type `hashes`.

```
"hashes": {
    "SHA-256": "effb46bba03f6c8aea5c653f9cf984f170dcdd3bbbe2ff6843c3e5da0e698766",
    "MD5": "9e04af713d91d493ef3301a050a18b7a"
    "SHA-1": "8bd560c15248aa8a2473d6fdbd0e83f202c891a9"
  },
```

  The filter `?match[MD5]=9e04af713d91d493ef3301a050a18b7a` or the filter `?match[SHA-1]=8bd560c15248aa8a2473d6fdbd0e83f202c891a9` will return the associated X.509 Certificate object.

- **String type**: Although spaces are not allowed on either side of a comma separating multiple values in a filter, properties of type string can be filtered, even if the string contains white space. For example, the filter, `?match[subject]=please open me,happy birthday` is valid.

  As mentioned above special characters such as white space **MUST** be encoded. So, as an example "%20" corresponds to the space character, so the match filter example in the previous paragraph would be encoded as `?match[subject]=please%20open%20me,happy%20birthday`.

  String matching is case-insensitive.

Additional match fields are shown in the subsections below. Tiered match fields should be implemented sequentially, but the relationships and calculation match fields can be implemented independently.

Tiered Match Fields

Property-based match fields have been divided into three tiers based on the structure of STIX 2.1. Match fields are alphabetized within each tier.

- **Tier 1**: match fields correspond to simple top-level properties of STIX objects.
- **Tier 2**: match fields correspond to array elements (lists) defined as top-level properties of STIX objects.
- **Tier 3**: match fields correspond to properties defined within nested structures.

Tier 1

Tier 1 match fields correspond to simple top-level properties of STIX objects. Properties with value type `identifier` that reference a relationship (end in "_ref") are defined in the Relationship class (see Relationships Match).

Match Field	Description

`account_type`	The type of **User Account** object.
	
	Value type: `account-type-ov`
	
	Examples
	`?match[account_type]=windows-local`
	`?match[account_type]=facebook,skype`

`confidence`	The **`confidence`** value applied to any STIX object(s).
	
	Value type: `integer`
	
	Examples
	`?match[confidence]=90`
	`?match[confidence]=90,91,92,93,94,95,96,97,98,99,100`

`context`	A short descriptor of the particular context shared by the content referenced by the **Grouping** object.
	
	Value type: `grouping-context-ov`
	
	Examples
	`?match[context]=suspicious-activity`
	`?match[context]=malware-analysis,unspecified`

`data_type`	The data type of the **Windows Registry Value** object.
	
	Value type: `windows-registry-datatype-enum`
	
	Examples
	`?match[data_type]=REG_BINARY`
	`?match[data_type]=REG_DWORD_BIG_ENDIAN,REG_DWORD_LITTLE_ENDIAN`

`dst_port`	The destination port used in a **Network Traffic** object.
	
	Value type: `integer`
	
	Examples
	`?match[dst_port]=1040`
	`?match[dst_port]=88841,83452`

`encryption_algorithm`	Specifies the type of encryption algorithm used to encode the binary data of an **Artifact** object.
	Value type: `encryption-algorithm-enum`
	
	Examples
	`?match[encryption_algorithm]=mime-type-indicated`
	`?match[encryption_algorithm]=AES-256-GCM,ChaCha20-Poly1305`

`identity_class`	The type of entity that an **Identity** object describes.
	
	Value type: `identity-class-ov`
	
	Examples
	`?match[identity_class]=individual`
	`?match[identity_class]=individual,group`

`name`	The name of objects (**Attack Pattern**, **Campaign**, **Course of Action**, **Grouping**, **Identity**, **Incident**, **Indicator**, **Infrastructure**, **Intrusion Set**, **Location**, **Malware**, **Report**, **Threat Actor**, **Tool**, **Vulnerability**, **Autonomous System (AS)**, **File**, **Mutex**, **Software**, **Marking Definition**, **Extension Definition**) and types (**Alternate Data Stream**, **Windows PE Section**, **Windows Registry Value**).
	
	Value type: `string`
	
	Examples
	`?match[name]=__CLEANSWEEP__`
	`?match[name]=Green%20Group%20Attackers,Panda%20Cubs%20United`

`number`	The number assigned to an **Autonomous System** object.
	
	Value type: `integer`
	
	Examples
	`?match[number]=15139`
	`?match[number]=19347,3954`

`opinion`	The opinion value present in an **Opinion** object.
	
	Value type: `opinion-enum`
	
	Examples
	`?match[opinion]=agree`
	
	`?match[opinion]=agree,strongly-agree`

`pattern`	The detection pattern for an **Indicator** object.
	
	Value type: `string`
	
	Examples
	`?match[pattern]=[file:hashes.'SHA-256' = '4bac27393bdd9777ce02453256c5577cd02275510b2227f473d03f533924f877']`
	`?match[pattern]=[file:hashes.MD5 = '3773a88f65a5e780c8dff9cdc3a056f3'],[file:hashes.'SHA-256' = 'ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c']`

`pattern_type`	The pattern language used in an **Indicator** object.
	
	Value type: `pattern-type-ov`
	
	Examples
	`?match[pattern_type]=stix`
	`?match[pattern_type]=sigma,snort`

`primary_motivation`	The primary reason, motivation, or purpose behind an **Intrusion Set** object or **Threat Actor** object.
	
	Value type: `attack-motivation-ov`
	
	Examples
	`?match[primary_motivation]=revenge`
	`?match[primary_motivation]=organization-gain,personal-gain`

`region`	The region a **Location** object describes.
	
	Value type: `region-ov`
	
	Examples
	`?match[region]=europe`
	`?match[region]=caribbean,south-america`

`relationship_type`	The type of relationship between the source and target objects in a **Relationship** object.
	
	Value type: `string`
	
	Examples
	`?match[relationship_type]=indicates`
	`?match[relationship_type]=indicates,uses`

`resource_level`	The organizational level at which an **Intrusion Set** object or **Threat Actor** object typically works, which determines the resources available for use in an attack.
	
	Value type: `attack-resource-level-ov`
	
	Examples
	`?match[resource_level]=government`
	`?match[resource_level]=team,organization`

`result`	The classification result of the **Malware Analysis** object as determined by the scanner or tool analysis process.
	
	Value type: `malware-result-ov`
	
	Examples
	`?match[result]=malicious`
	`?match[result]=benign,unknown`

`revoked`	Returns STIX objects based on the **revoked** property. The **revoked** property is optional and has specific interpretation in the STIX specification. For example, if the **revoked** property is not present, the object is considered valid (default is `false`).
	
	● ?match[revoked]=true will return objects that have been revoked (the **revoked** property equals `true`).
	● ?match[revoked]=false will return objects that have not been revoked (the **revoked** property is not present or equals `false`).
	
	Value type: `boolean`
	
	Examples
	`?match[revoked]=false`
	`?match[revoked]=true`

`src_port`	The source port used in a **Network Traffic** object.
	
	Value type: `integer`
	
	Examples
	`?match[src_port]=9081`
	`?match[src_port]=3372,24638`

`sophistication`	The skill, specific knowledge, special training, or expertise a **Threat Actor** object **MUST** have to perform an attack.
	
	Value type: `threat-actor-sophistication-ov`
	
	Examples
	`?match[sophistication]=none`
	`?match[sophistication]=expert,innovator`

`subject`	Specifies the subject of an **Email Message** or **X.509 Certificate** object.
	
	Value type: `string`
	
	Examples
	`?match[subject]=happy%20birthday`
	`?match[subject]=see%20this%20joke,funny%20photo`

`value`	The value present in STIX SCOs **ipv4-addr**, **ipv6-addr**, **domain-name**, **email-addr**, **mac-addr**, and **url** objects value property.
	
	Value type: `string`
	
	Examples
	`?match[value]=198.51.100.3`
	`?match[value]=john@example.com,doe@example.com`

Tier 2

Tier 2 match fields correspond to array elements (lists) defined as top-level properties of STIX objects. Properties with value type `identifier` that reference relationships (end in "_refs") are defined in the Relationships class.

Match Field	Description

`aliases`	Alternative names used to identify **Attack Pattern**, **Campaign**, **Infrastructure**, **Intrusion Set**, **Malware**, **Threat Actor**, and **Tool** objects.
	Value type: `string`
	Examples
	`?match[aliases]=Zookeeper`
	`?match[aliases]=Syndicate%201,Evil%20Syndicate%2099`

`architecture_`
`executions_envs`
	The processor architectures that **Malware** object is executable on.
	Value type: `processor-architecture-ov`
	Examples
	`?match[architecture_executions_envs]=x86`
	`?match[architecture_executions_envs]=x86,x86-64`

`capabilities`	The capabilities of **Malware** object.
	Value type: `malware-capabilities-ov`
	Examples
	`?match[capabilities]=emails-spam`
	`?match[capabilities]=anti-debugging,anti-disassembly`

`extension_types`	The type of the **Extension** meta-object.
	Value type: `extension-type-enum`
	Examples
	`?match[extension_types]=new-sdo`
	`?match[extension_types]=new-sdo,new-sco`

`implementation_`
`languages`
	The programming language used to implement **Malware** object.
	Value type: `implementation-language-ov`
	Examples
	`?match[implementation_languages]=visual-basic`
	`?match[implementation_languages]=java,php`

`indicator_types`	The category of an **Indicator** object.
	
	Value type: `indicator-type-ov`
	
	Examples
	`?match[indicator_types]=anonymization`
	`?match[indicator_types]=compromised,malicious-activity`

`infrastructure_types`	The type of **Infrastructure** object.
	
	Value type: `infrastructure-type-ov`
	
	Examples
	`?match[infrastructure_types]=botnet`
	`?match[infrastructure_types]=phishing,reconnaissance`

`labels`	The label value(s) applied to any STIX object.
	
	Value type: `string`
	
	Examples
	`?match[labels]=trickbot`
	`?match[labels]=totbrick,tspy_trickload`

`malware_types`	The category of **Malware** object.
	
	Value type: `malware-type-ov`
	
	Examples
	`?match[malware_types]=bot`
	`?match[malware_types]=virus,worm`

`personal_motivations`	The personal reasons, motivations, or purposes of a **Threat Actor** object, regardless of organizational goals.
	
	Value type: `attack-motivation-ov`
	
	Examples
	`?match[personal_motivations]=accidental`
	`?match[personal_motivations]=ideology,notoriety`

`report_types`	The primary type of content found in a **Report** object.
	
	Value type: `report-type-ov`
	
	Examples
	`?match[report_types]=indicator`
	`?match[report_types]=malware,tool`

`roles`	The roles performed by the **Identity** object.
	
	Value type: `string`
	
	Examples
	`?match[labels]=ceo`
	`?match[labels]=doctor,hospital`

`roles`	The roles played by a **Threat Actor** object.
	Value type: `threat-actor-role-ov`
	Examples
	`?match[roles]=malware-author`
	`?match[roles]=agent,director`

`secondary_motivations`	The secondary reasons, motivations, or purposes behind an **Intrusion Set** object or **Threat Actor** object.
	Value type: `attack-motivation-ov`
	Examples
	`?match[secondary_motivations]=ideology`
	`?match[secondary_motivations]=dominance,revenge`

`sectors`	The sectors property defined in an **Identity** object.
	Value type: `industry-sector-ov`
	Examples
	`?match[sectors]=energy`
	`?match[sectors]=financial-services,manufacturing`

`threat_actor_types`	The type of **Threat Actor** object.
	Value type: `threat-actor-type-ov`
	Examples
	`?match[threat_actor_types]=criminal`
	`?match[threat_actor_types]=nation-state,terrorist`

`tool_types`	The type of **Tool** object.
	Value type: `tool-type-ov`
	Examples
	`?match[capabilities]=network-capture`
	`?match[capabilities]=credential-exploitation,remote-access`

Tier 3

Tier 3 match fields correspond to properties defined within nested structures.

Match Field	Description

`address_family`	The address family of the **Network Socket** object.
	Value type: `network-socket-address-family-enum`
	Examples
	`?match[address_family]=AF_APPLETALK`
	`?match[address_family]=AF_INET,AF_INET6`

`external_id`	An identifier present in any STIX object(s) **`external_references`** property.
	
	Value type: `string`
	
	Examples
	`?match[external_id]=CVE-2016-1234`
	`?match[external_id]=CWE-20,T1245`

Hashes
 	
`MD5`
`SHA-1`
`SHA-256`
`SHA-512`
`SHA3-256`
`SHA3-512`
`SSDEEP`
`TLSH`
	The Hashing Algorithm open vocabulary (`hash-algorithm-ov`) is used in the **External Reference**, **Artifact**, **File**, **Alternate Data Stream**, **Windows PE Binary File**, **Windows PE Optional Header**, **Windows PE Section**, and **X.509 Certificate** objects, which each include a hashes property of type `hashes` (a set of key/value pairs).
	
	Value type: `string`
	
	Examples
	`?match[SHA-256]=35a01331e9ad96f751278b891b6ea09699806faedfa237d40513d92ad1b7100f`
	
	`?match[MD5]=9e04af713d91d493ef3301a050a18b7a,53d780fc1453f56d6dff77a93a920794`

`integrity_level`	The integrity level of the **Windows Process** object.
	
	Value type: `windows-integrity-level-enum`
	
	Examples
	`?match[integrity_level]=high`
	`?match[integrity_level]=medium,high`

`pe_type`	The type of **PE** binary object.
	
	Value type: `windows-pebinary-type-ov`
	
	Examples
	`?match[pe_type]=dll`
	`?match[pe_type]=dll,exe`

`phase_name`	The name of the phase in a kill chain as defined in the **`kill_chain_phases`** property of an Attack Pattern, Indicator, Infrastructure, Malware, or Tool object.
	
	Value type: `string`
	
	Examples
	`?match[phase_name]=reconnaissance`
	`?match[phase_name]=pre%2Dattack,post%2Dattack`

`service_status`	The current status of the **Windows Service** object.
	
	Value type: `windows-service-status-enum`
	
	Examples
	`?match[service_status]=SERVICE_STOPPED`
	`?match[service_status]=SERVICE_RUNNING,SERVICE_START_PENDING`

`service_type`	The type of the **Windows Service** object.
	Value type: `windows-service-type-enum`
	Examples
	`?match[service_type]=SERVICE_WIN32_OWN_PROCESS`
	`?match[service_type]=SERVICE_KERNEL_DRIVER,SERVICE_FILE_SYSTEM_DRIVER`

`socket_type`	The type of **Network Socket** object.
	Value type: `network-socket-type-enum`
	Examples
	`?match[socket_type]=SOCK_RAW`
	`?match[socket_type]=SOCK_STREAM,SOCK_SEQPACKET`

`source_name`	A source name present in any STIX object(s) **`external_references`** property.
	Value type: `string`
	Examples
	`?match[source_name]=cve`
	`?match[source_name]=capec,veris`

`start_type`	The start options of the **Windows Service** object.
	Value type: `windows-service-start-type-enum`
	Examples
	`?match[start_type]=SERVICE_DISABLED`
	`?match[start_type]=SERVICE_AUTO_START,SERVICE_BOOT_START`

`tlp`	The marking-definition identifier applied to object(s). This is a shorthand to objects specifically marked with a TLP marking. The only allowed values **MUST** are `white`, `green`, `amber`, and `red`. Specific IDs for each TLP color **MUST** be mapped as defined on the **TLP Marking Object Type** section in [STIX™ Version 2.1].
	Value type: `string`
	Examples
	`?match[tlp]=white`
	`?match[tlp]=white,green`

Relationships Match Field

The Relationships class consists of a `relationships-all` match field that matches against any property that ends in "`_ref`" or "`_refs`" (value type `identifier`). For example, the filter

```
?match[relationships-all]=indicator--3600ad1b-fff1-4c98-bcc9-4de3bc2e2ffb
```

will return all objects that reference the given indicator object.

STIX object properties relevant to the `relationships-all` match field, as well as their descriptions, are given in the table below.

Property	Description
`analysis_sco_refs`	Specifies the SCO captured during the analysis process of a **Malware Analysis** object.
`bcc_refs`	Specifies the mailboxes that are "BCC" recipients of an **Email Message** object.
`belongs_to_ref`	Specifies the user account that the **Email Address** object belongs to.
`belongs_to_refs`	Specifies one or more autonomous systems that the **IPv4 Address** or **IPv6 Address** object belongs to.
`body_raw_ref`	Specifies the contents of non-textual MIME parts of an **Email MIME Component Type** object.
`cc_refs`	Specifies the mailboxes that are "CC" recipients of an **Email Message** object.
`child_refs`	Specifies the other processes that were spawned by the **Process** object.
`contains_refs`	Specifies other files or directory objects contained in a **Directory** or **Archive File Extension** object. Specifies other SCOs contained in a **File** object.
`content_ref`	Specifies the content of a **File** object.
`created_by_ref`	Specifies the identity creator identifier applied to any STIX object(s).
`creator_user_ref`	Specifies the user account that created the **Process** or **Windows Registry Key** object.
`dst_payload_ref`	Specifies the bytes sent from the destination to source in a **Network Traffic** object.
`dst_ref`	Specifies the destination of a **Network Traffic** object.
`encapsulated_by_ref`	Specifies a network traffic object that encapsulate a **Network Traffic** object.
`encapsulates_refs`	Specifies other network traffic objects encapsulated by a **Network Traffic** object.
`from_ref`	Specifies the value of the "From" header of an **Email Message** object. The "From" field specifies the author of the message (i.e., the mailbox of the person or system responsible for the writing of the message).

`host_vm_ref`	Specifies the virtual machine (software) environment used by a **Malware Analysis** object.
`image_ref`	Specifies the executable binary that was executed as the process image by a **Process** object.
`installed_software_refs`	Specifies software used by a **Malware Analysis** object.
`marking_ref`	Specifies the marking definition that describes a **Granular Marking** type.
`message_body_data_ref`	Specifies the data contained in an **HTTP Request Extension** object.
`object_marking_refs`	Specifies the marking definition applied to any STIX object.
`object_ref`	Specifies the object that the **Language Content** object applies to.
`object_refs`	Specifies the objects referred to by a **Grouping**, **Note**, **Observed Data**, **Opinion** or **Report** object.
`observed_data_refs`	Specifies the raw cyber data for a **Sighting** object.
`opened_connections_refs`	Specifies the network connections opened by a **Process** object.
`operating_system_ref`	Specifies the operating system used for analysis in a **Malware Analysis** object.
`operating_system_refs`	Specifies the operating systems that a **Malware** object executes on.
`parent_directory_ref`	Specifies the parent directory of a **File** object.
`parent_ref`	Specifies the process that spawned a **Process** object.
`raw_email_ref`	Specifies the raw binary contents of an **Email Message** object.
`resolves_to_refs`	Specifies an IPv4 address, IPv6 address, or domain name that a **Domain Name** object resolves to. May also specify a MAC address that an **IPv4 Address** or **IPv6 Address** object resolves to.
`sample_ref`	Specifies a file, network traffic, or artifact object that the **Malware Analysis** object was performed against.
`sample_refs`	Specifies a file or artifact object associated with a **Malware** object.

`sender_ref`	The value of the "Sender" field of an **Email Message** object. The "Sender" field specifies the mailbox of the agent responsible for the actual transmission of the message.
`service_dll_refs`	Specifies the DLLs loaded by a **Windows Service Extension** object.
`sighting_of_ref`	Specifies the SDO referenced in a **Sighting** object.
`source_ref`	Specifies the source SDO or SCO contained in a **Relationship** object.
`src_payload_ref`	Specifies the bytes sent from the source to the destination in a **Network Traffic** object.
`src_ref`	Specifies the source of a **Network Traffic** object.
`target_ref`	Specifies the target SDO or SCO contained in a **Relationship** object.
`to_refs`	Specifies the mailboxes that are "To" recipients of an **Email Message** object.
`where_sighted_refs`	Specifies the identities or locations describing the entities that saw a **Sighting** object.

Calculation Match Field

The Calculation class defines match fields that require calculation, rather than a simple match.

Match Field	Description
`confidence-gte`	Returns STIX objects with **`confidence`** property values greater than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the smallest value. **Example** `?match[confidence-gte]=80`
`confidence-lte`	Returns STIX objects with **`confidence`** property values less than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the largest value. **Example** `?match[confidence-lte]=70`

`modified-gte`	Returns STIX objects that have a **`modified`** property that is on or after a specific timestamp. A filter **SHOULD** contain only a single timestamp. If multiple timestamps are provided, the filter is equivalent to using only the earliest timestamp. **Example** `?match[modified-gte]=2021-01-05T12:10:01.000Z`
`modified-lte`	Returns STIX objects that have a **`modified`** property that is on or before a specific timestamp. A filter **SHOULD** contain only a single timestamp. If multiple timestamps are provided, the filter is equivalent to using only the latest timestamp. **Example** `?match[modified-lte]=2021-06-27T00:00:00.000Z`
`number-gte`	Returns **Autonomous System** objects where the **`number`** property is greater than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the smallest value. **Example** `?match[number-gte]=15000`
`number-lte`	Returns **Autonomous System** objects where the **`number`** property is less than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the largest value. **Example** `?match[number-lte]=7500`
`src_port-gte`	Returns **Network Traffic** objects where the **`src_port`** property is greater than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the smallest value. **Example** `?match[src_port-gte]=5000`
`src_port-lte`	Returns **Network Traffic** objects where the **`src_port`** property is less than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the largest value. **Example** `?match[src_port-lte]=22000`

`dst_port-gte`	Returns **Network Traffic** objects where the **`dst_port`** property is greater than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the smallest value. **Example** `?match[dst_port-gte]=9500`
`dst_port-lte`	Returns **Network Traffic** objects where the **`dst_port`** property is less than or equal to a given value. A filter **SHOULD** contain only a single value. If multiple values are provided, the filter is equivalent to using only the largest value. **Example** `?match[dst_port-lte]=2000`
`valid_until-gte`	Returns **Indicator** objects that have a **`valid_until`** property that is on or after a specific timestamp. A filter **SHOULD** contain only a single timestamp. If multiple timestamps are provided, the filter is equivalent to using only the earliest timestamp. The **`valid_until`** property is optional and has specific interpretation in the STIX specification. For example, if the **`valid_until`** property is not present, the object is considered valid. Therefore, an indicator without a **`valid_until`** property will be returned. **Example** `?match[valid_until-gte]=2021-09-01T12:05:00.000Z`
`valid_from-lte`	Returns Indicator objects that have a **`valid_from`** property that is on or before a specific timestamp. A filter **SHOULD** contain only a single timestamp. If multiple timestamps are provided, the filter is equivalent to using only the earliest timestamp. **Example** `?match[valid_from-lte]=2020-05-25T01:01:01.000Z`

Appendix C. Acknowledgments

Interoperability Subcommittee Chairs

Stephen Russett, Cyber Threat Intelligence Network, Inc.
Marlon Taylor, DHS
Michael Rosa, DHS
Jason Keirstead, IBM
Allan Thomson, Individual
Justin Stewart, LookingGlass
Rajesh Patil, LookingGlass
Kartikey Desai, MITRE Corporation

Special Thanks

Substantial contributions to this specification from the following individuals are gratefully acknowledged:

Christian Hunt, Copado
Bret Jordan, Cyber Threat Intelligence Network, Inc.
Jane Ginn, Cyber Threat Intelligence Network, Inc.
Christopher Robinson, Cyber Threat Intelligence Network, Inc.
Jeffrey Mates, US Department of Defense (DoD)
Keven Ates, US Federal Bureau of Investigation (FBI)
Ryusuke Masuoka, Fujitsu Limited
Scott Robertson, Kaiser Permanente
Gus Creedon, Logistics Management Institute
Chris Lenk, MITRE Corporation
Alexandre Cabrol Perales, SEKOIA

Participants

The following individuals were members of the OASIS CTI Technical Committee during the creation of this specification.

First Name	**Last Name**	**Organization**
Robert	Coderre	Accenture
Robert	Keith	Accenture
Curtis	Kostrosky	Accenture
Kyle	Maxwell	Accenture
Florian	Skopik	AIT Austrian Institute of Technology

Scott	Dowsett	Anomali
Wei	Huang	Anomali
Russell	Matbouli	Anomali
Hugh	Njemanze	Anomali
Katie	Pelusi	Anomali
Patrick	Maroney	AT&T
Dean	Thompson	Australia and New Zealand Banking Group (ANZ Bank)
Radu	Marian	Bank of America
Charles	Yarbrough	Carnegie Mellon University
Trey	Darley	CCB/CERT.be
Alexandre	Dulaunoy	CIRCL
Andras	Iklody	CIRCL
Christian	Studer	CIRCL
Raphaël	Vinot	CIRCL
Syam	Appala	Cisco Systems
Ted	Bedwell	Cisco Systems
Caitlin	Huey	Cisco Systems
Pavan	Reddy	Cisco Systems
Omar	Santos	Cisco Systems
Sam	Taghavi Zargar	Cisco Systems
Jyoti	Verma	Cisco Systems
Andrew	Windsor	Cisco Systems
Kevin	Chan	Copado

Kelly	Cullinane	Copado
John-Mark	Gurney	Copado
Christian	Hunt	Copado
Daniel	Riedel	Copado
Andrew	Storms	Copado
Tim	Hudson	Cryptsoft Pty Ltd.
Arsalan	Iqbal	CTM360
Jane	Ginn	Cyber Threat Intelligence Network, Inc. (CTIN)
Bret	Jordan	Cyber Threat Intelligence Network, Inc. (CTIN)
Ben	Ottoman	Cyber Threat Intelligence Network, Inc. (CTIN)
David	Powell	Cyber Threat Intelligence Network, Inc. (CTIN)
Christopher	Robinson	Cyber Threat Intelligence Network, Inc. (CTIN)
Andreas	Sfakianakis	Cyber Threat Intelligence Network, Inc. (CTIN)
Nick	Sturgeon	Cyber Threat Intelligence Network, Inc. (CTIN)
Michael	Butt	Cyware Labs
Utkarsh	Garg	Cyware Labs
Anuj	Goel	Cyware Labs
Avkash	Kathiriya	Cyware Labs
Andrew	Nau	Cyware Labs
Timothy	Casey	DarkLight, Inc.
Ryan	Hohimer	DarkLight, Inc.
Ryan	Joyce	DarkLight, Inc.
Paul	Patrick	DarkLight, Inc.

Andrew	Byrne	Dell
Ravi	Sharda	Dell
Will	Urbanski	Dell
David	Ailshire	DHS Office of Cybersecurity and Communications (CS&C)
Steven	Fox	DHS Office of Cybersecurity and Communications (CS&C)
Taneika	Hill	DHS Office of Cybersecurity and Communications (CS&C)
Evette	Maynard-Noel	DHS Office of Cybersecurity and Communications (CS&C)
Jackie	Eun Park	DHS Office of Cybersecurity and Communications (CS&C)
Sean	Sobieraj	DHS Office of Cybersecurity and Communications (CS&C)
Marlon	Taylor	DHS Office of Cybersecurity and Communications (CS&C)
Preston	Werntz	DHS Office of Cybersecurity and Communications (CS&C)
Joep	Gommers	EclecticIQ
Sergey	Polzunov	EclecticIQ
Zed	Tan	EclecticIQ
Aukjan	van Belkum	EclecticIQ
Raymon	van der Velde	EclecticIQ
Joseph	Woodruff	EclecticIQ
Ben	Sooter	Electric Power Research Institute (EPRI)
Chris	Ricard	Financial Services Information Sharing and Analysis Center (FS-ISAC)
Ryusuke	Masuoka	Fujitsu Limited
Toshitaka	Satomi	Fujitsu Limited
Koji	Yamada	Fujitsu Limited
Robert	van Engelen	Genivia

Mark	Risher	Google Inc.
Yoshihide	Kawada	Hitachi, Ltd.
Jun	Nakanishi	Hitachi, Ltd.
Kazuo	Noguchi	Hitachi, Ltd.
Akihito	Sawada	Hitachi, Ltd.
Yutaka	Takami	Hitachi, Ltd.
Masato	Terada	Hitachi, Ltd.
Eldan	Ben-Haim	IBM
Roseann	Guttierrez	IBM
Sandra	Hernandez	IBM
Jason	Keirstead	IBM
Chenta	Lee	IBM
John	Morris	IBM
Emily	Ratliff	IBM
Aviv	Ron	IBM
Nick	Rossmann	IBM
Laura	Rusu	IBM
Sulakshan	Vajipayajula	IBM
Ron	Williams	IBM
Joerg	Eschweiler	Individual
Elysa	Jones	Individual
Terry	MacDonald	Individual
Anthony	Rutkowski	Individual

Allan	Thomson	Individual
James	Cabral	InfoTrack US
Jorge	Aviles	Johns Hopkins University Applied Physics Laboratory
Julie	Modlin	Johns Hopkins University Applied Physics Laboratory
Mark	Moss	Johns Hopkins University Applied Physics Laboratory
Mark	Munoz	Johns Hopkins University Applied Physics Laboratory
Nathan	Reller	Johns Hopkins University Applied Physics Laboratory
Pamela	Smith	Johns Hopkins University Applied Physics Laboratory
Russell	Culpepper	Kaiser Permanente
Michael	Slavick	Kaiser Permanente
Kent	Landfield	McAfee
Desiree	Beck	Mitre Corporation
Michael	Chisholm	Mitre Corporation
Mike	Cokus	Mitre Corporation
Sam	Cornwell	Mitre Corporation
Kartikey	Desai	Mitre Corporation
Danny	Haynes	Mitre Corporation
Chris	Lenk	Mitre Corporation
Nicole	Parrish	Mitre Corporation
Richard	Piazza	Mitre Corporation
Larry	Rodrigues	Mitre Corporation
Zach	Rush	Mitre Corporation
Jon	Salwen	Mitre Corporation

Richard	Struse	Mitre Corporation
Alex	Tweed	Mitre Corporation
Emmanuelle	Vargas-Gonzalez	Mitre Corporation
John	Wunder	Mitre Corporation
Scott	Algeier	National Council of ISACs (NCI)
Denise	Anderson	National Council of ISACs (NCI)
Josh	Poster	National Council of ISACs (NCI)
Mike	Boyle	National Security Agency
Jessica	Fitzgerald-McKay	National Security Agency
David	Kemp	National Security Agency
Shaun	McCullough	National Security Agency
Daichi	Hasumi	NEC Corporation
Takahiro	Kakumaru	NEC Corporation
Lauri	Korts-Pärn	NEC Corporation
Drew	Varner	NineFX, Inc.
Stephen	Banghart	NIST
James	Crossland	Northrop Grumman
Robert	Van Dyk	Northrop Grumman
Cheolho	Lee	NSR
Joel	Myhre	Pacific Disaster Center
Stephan	Relitz	Peraton
David	Bizeul	SEKOIA
Georges	Bossert	SEKOIA

Duncan	Sparrell	sFractal Consulting LLC
Marco	Caselli	Siemens AG
Alexandre	Cabrol Perales	Sopra Steria Group
Margaux	Quittelier	Sopra Steria Group
Adam	Wyner	Swansea University
Srujan	Kotikela	Texas A&M University-Commerce
Andrew	Gidwani	ThreatConnect, Inc.
Cole	Iliff	ThreatConnect, Inc.
Andrew	Pendergast	ThreatConnect, Inc.
Jason	Spies	ThreatConnect, Inc.
Jason	Avery	Trend Micro
Ed	Cabrera	Trend Micro
ziv	chang	Trend Micro
David	Girard	Trend Micro
Brandon	Niemczyk	Trend Micro
Jessie	Chuang	TWNCERT
Julie	Wang	TWNCERT
Vasileios	Mavroeidis	University of Oslo
Mateusz	Zych	University of Oslo
Jeffrey	Mates	US Department of Defense (DoD)
Keven	Ates	US Federal Bureau of Investigation
Shaun	McCullough	National Security Agency
Michael	Rosa	National Security Agency

Daichi	Hasumi	NEC Corporation
Takahiro	Kakumaru	NEC Corporation
Lauri	Korts-Pärn	NEC Corporation
Drew	Varner	NineFX, Inc.
Stephen	Banghart	NIST
Scott	Carlisle	Northrop Grumman
James	Crossland	Northrop Grumman
Ivan	Diaz	Northrop Grumman
Anthony	Lay	Northrop Grumman
Qem	Lumi	Northrop Grumman
Robert	Van Dyk	Northrop Grumman
Cheolho	Lee	NSR
James Bryce	Clark	OASIS
Chet	Ensign	OASIS
Web	Master	OASIS
CTI	Mirror	OASIS
cti-cybox	Mirror	OASIS
cti-stix	Mirror	OASIS
cti-taxii	Mirror	OASIS
Dee	Schur	OASIS
Patrick	Bredenberg	Oracle
Johnny	Gau	Oracle
Sunil	Ravipati	Oracle

Joel	Myhre	Pacific Disaster Center
Ryan	Clough	Palo Alto Networks
Ryan	Olson	Palo Alto Networks
Jason	Liu	Peraton
Stephan	Relitz	Peraton
Altaz	Valani	Security Compass
David	Bizeul	SEKOIA
Georges	Bossert	SEKOIA
Duncan	Sparrell	sFractal Consulting LLC
Marco	Caselli	Siemens AG
Jonas	Plum	Siemens AG
Jeremy	Berthelet	Sopra Steria Group
Alexandre	Cabrol Perales	Sopra Steria Group
Adam	Wyner	Swansea University
Alan	Steer	TELUS
Srujan	Kotikela	Texas A&M University-Commerce
Andrew	Gidwani	ThreatConnect, Inc.
Cole	Iliff	ThreatConnect, Inc.
Andrew	Pendergast	ThreatConnect, Inc.
Jason	Spies	ThreatConnect, Inc.
Alejandro	Valdivia	ThreatConnect, Inc.
Haig	Colter	ThreatQuotient, Inc.
Jason	Avery	Trend Micro

Ed	Cabrera	Trend Micro
Ziv	Chang	Trend Micro
David	Girard	Trend Micro
Robert	McArdle	Trend Micro
Brandon	Niemczyk	Trend Micro
Jessie	Chuang	TWNCERT
Julie	Wang	TWNCERT
Vasileios	Mavroeidis	University of Oslo
Ulrik	Palmstrøm	University of Oslo
Jeffrey	Mates	US Department of Defense (DoD)
Keven	Ates	US Federal Bureau of Investigation

Appendix D. Revision History

Revision	Date	Editor	Changes Made
01	2022-03-01	Dez Beck Kartikey Desai Marlon Taylor	Initial version. Updated use cases to use TAXII 2.1, add new use cases, refreshed personas, and added advanced filtering.