

# IUA

34 Internet User Authorization (IUA) Profile

## Integrating the Healthcare Enterprise



[IHE ITI](#)

### Technical Framework Supplement

### Internet User Authorization (IUA)

### Revision 2.1 - Trial Implementation

Date: March 19, 2021

Author: ITI Technical Committee

Email: [iti@ihe.net](mailto:iti@ihe.net)

**Please verify you have the most recent version of this document.** See [here](#) for Trial Implementation and Final Text versions and [here](#) for Public Comment versions.

### Foreword

This is a supplement to the IHE IT Infrastructure Technical Framework. Each supplement undergoes a process of public comment and trial implementation before being incorporated into the volumes of the Technical Frameworks.

This supplement is published on March 19, 2021 for trial implementation and may be available for testing at subsequent IHE Connectathons. The supplement may be amended based on the results of testing. Following successful testing it will be incorporated into the IT Infrastructure Technical Framework. Comments are invited and can be submitted using the [ITI Public Comment form](#) or by creating a [GitHub Issue](#).

This supplement describes changes to the existing technical framework documents.

"Boxed" instructions like the sample below indicate to the Volume Editor how to integrate the relevant section(s) into the relevant Technical Framework volume.

**Editor: Please amend Section X.X by the following**

Where the amendment adds text, make the added text **bold underline**. Where the amendment removes text, make the removed text ~~bold strikethrough~~. When entire new sections are added, introduce with editor's instructions to "add new text" or similar, which for readability are not bolded or underlined.

General information about IHE can be found at <http://www.ihe.net>.

Information about the IHE IT Infrastructure domain can be found at [https://www.ihe.net/IHE\\_Domains](https://www.ihe.net/IHE_Domains).

Information about the organization of IHE Technical Frameworks and Supplements and the process used to create them can be found at [https://www.ihe.net/about\\_ihe/ihe\\_process](https://www.ihe.net/about_ihe/ihe_process) and <https://www.ihe.net/resources/profiles>.

The current version of the IHE Technical Framework can be found at <https://profiles.ihe.net/>.

### CONTENTS

- [Introduction to this Supplement](#)
- [Open Issues and Question](#)
- [Closed Issues](#)
- [IHE Technical Frameworks General Introduction](#)
  - [9 Copyright Licenses](#)
- [IHE Technical Frameworks General Introduction Appendices](#)
  - [Appendix A - Actor Summary Definitions](#)
  - [Appendix B - Transaction Summary Definitions](#)
  - [Appendix D - Glossary](#)
- [Volume 1 -- Profiles](#)
- [34 IUA Profile](#)
  - [34.1 IUA Actors, Transactions, and Content Modules](#)
  - [34.2 IUA Actor Options](#)
  - [34.3 IUA Required Actor Groupings](#)
  - [34.4 IUA Overview](#)
  - [34.5 IUA Security Considerations](#)
  - [34.6 IUA Cross Profile Considerations](#)
- [Volume 2 -- Transactions](#)
  - [3.71 Get Access Token \[ITI-71\]](#)
  - [3.72 Incorporate Access Token \[ITI-72\]](#)
  - [3.102 Introspect Token \[ITI-102\]](#)
  - [3.103 Get Authorization Server Metadata \[ITI-103\]](#)
- [33 MHD Profile](#)
- [9 ATNA Profile](#)

## Introduction to this Supplement

### Problem Statement

This profile is motivated by customer requirements for authorizing network transactions when using HTTP RESTful transports. IHE has authorization profiles for the Web Services and SOAP based transactions, and this profile provides an authorization profile for the HTTP RESTful transactions.

Being authorized means that the user, patient, or provider has legitimate access to this HTTP RESTful service. The authorization includes identifying the user and the application that is making the request to the HTTP RESTful server, so that server can make further access control decisions.

The HTTP RESTful transport is being used by many healthcare applications and smart devices. These share a common set of issues. A typical use case example is:

- The patient installs an application onto his tablet.
- The application will need to retrieve and update health related data that is stored on a Resource Server. It uses HTTP RESTful transactions for both retrieve and update because HTTP support is integrated into the platform services.

- The patient wants to configure the application to have access to his data without needing the IT staff at the application vendor and resource vendor to set things up.

The HTTP RESTful services may include user-driven browser activity, downloaded applications, automatic devices, and IT systems. The existing IHE ITI [Cross-Enterprise User Assertion \(XUA\)](#) Profile fills these needs for the SOAP transport based transactions. The existing IHE ITI [Enterprise User Authentication \(EUA\)](#) Profile fills these needs for various different transports within a single enterprise environment, including HTTP RESTful transports.

The [Basic Patient Privacy Consent \(BPPC\)](#) Profile is associated with this profile and these other existing profiles. [BPPC](#) covers the legal and administrative needs for consent documentation and associating the patient consent with policy documentation. The [BPPC](#) Profile includes the ability to associate the electronic authorizations with the patient agreements and organizational policies.

IUA is not a substitute for the administrative activities such as withdrawing consent, policy setting, and other activities that [BPPC](#) documents provide. The current scope of IUA is the basic authorization activities. Future scope may include greater integration of this authorization with third party authorization and consent documentation profiles, such as those found in the IHE [BPPC](#) Profile.

The administrative actions needed to establish an Authorization Server for IUA are not covered by this profile. These activities depend upon the operational needs, and organizational and privacy policies that apply to a particular deployment.

## Background on the problem environment

The application interacts with the user and the Authorization Server to support the granting of an access token. The application then uses the access token to retrieve and update health related data.

The key issues include:

- Making reliable and accurate authorization decisions as part of an overall privacy protecting and security environment.
- Application developers expecting one common method for obtaining and using access tokens.
- Resource Servers expecting one common method for receiving access tokens as part of HTTP RESTful transactions, and one common method for processing these tokens.
- Users, patients, and providers, expecting to be in control, do not want to depend on support staff to set up their applications, and want to minimize the interference from authorization requirements.

Additional issues:

- In-house application distribution needing to provide authorization for applications used within the facility.
- IT administrators preferring a common method to authorize use of in-house web applications and access to in-house resources.
- Providers and Specialists having authorization needs to deal with other organizations and with many resource services.
- Granting subset access to specialized provider. For example, granting a physical therapy organization read access to a patient's cardiac information but forbidding access to other sensitive health data.

There are also environmental assumptions made by this profile.

IUA assumes that there will many systems working together to build the access control environment, where IUA provides one part only. For example, a glucose monitor may be authorized to have access to a patient's medical record. The expectation is that this will mean access to all of the glucose-related information, which will include a variety of measurements and prescriptions. But, it is expected that if the device requests other sensitive health data about the patient, it will be rejected.

IUA also assumes that this profile is operating in an environment where access consents are managed by [BPPC](#) or other mechanisms. For example, there will be a documented consent agreement between a patient and a provider that the provider will provide medical records to a healthcare proxy that is identified and authorized by the patient. [BPPC](#) is one way to document that agreement. IUA is not a substitute for documenting, establishing, and modifying these legal agreements.

## Relation to SMART-on-FHIR

IUA is not based on SMART-on-FHIR, but does strive to not conflict with that standard.

Concerns about SMART were raised surrounding the tight dependency between Resource and Authorization Server, the inclusion of launch/view context in the authorization flow, and the dependency on FHIR. The IHE IT Infrastructure Domain believes that IUA provides the following advantages over SMART:

- IUA promotes a loose coupling of Resource Server and Authorization Servers. This allows for deployments with multiple Resource Servers per Authorization Server as well as deployments with several or even no Authorization Servers.
- IUA supports a wide range of use-cases ranging from mobile application access to data, cross-enterprise data exchange to complex system integration scenarios.
- IUA is base-standard agnostic and can be combined with any HTTP RESTful transaction.
- IUA provides explicit means of obtaining access token claims from an access token by a resource server (with and without the involvement of an Authorization Server).
- IUA specifies additional authorization context claims such as [BPPC](#) consents and a user's organizational context.
- IUA provides explicit compatibility with IHE XUA.

That said, it is recognized that SMART-on-FHIR is evolving and adoption rates are increasing. Therefore, as indicated in Issue 13 below, IHE ITI welcomes community feedback on the harmonization of IUA and SMART.

## Open Issues and Questions

- **Issue 11:** IUA does not define scopes at all; therefore, IUA is not in conflict with SMART-on-FHIR. However, this supplement includes updates to [MHD](#) and does define a Scope for use with [MHD](#) transactions. That Scope for [MHD](#) does not use a SMART pattern, but it does not forbid SMART scopes. IHE requests comments on the IUA, MHD, and SMART-on-FHIR Scope harmonization.
- **Issue 12:** Given that the IUA Resource Server is grouped with some other IHE-defined actor, and that actor has audit logging requirements, IUA does not need to give a defined audit event for success. IUA does require adding an AuditEvent.entity to the AuditEvent defined in the other profile. IHE requests comments on a need to define in IUA an AuditEvent specific to a Resource Server that is enforcing a DENY. This AuditEvent would be used when the Resource Server prevented the transaction from reaching the grouped Profile/Actor (e.g., [MHD](#) Document Responder). This AuditEvent would cover reasons for DENY that are not specific to the content of the grouped transaction, e.g., reasons such as: Missing Token, Token validation failure, Token expiration, Scope mismatch, IUA required attributes missing, etc. The expectation is that we can leverage some codes from OAuth.
- **Issue 13:** The SMART-on-FHIR standard is gaining world-wide adoption and increases in importance. IUA does not utilize SMART-on-FHIR as base-standard, but refers to OAuth2.1 and FHIR directly. It is recognized that IUA and SMART-on-FHIR do have a (partial) overlap in the supported use-cases. IHE requests comments on IUA and SMART-on-FHIR flow harmonization.
- **Issue 14:** The IETF OAuth Working group developed an extension to the OAuth framework that specifies an option to convey more fine grained claims with the authorization request. The OAuth 2.0 Rich Authorization Requests (draft-ietf-oauth-rar-03) specification introduces a new parameter "authorization\_details" in authorization requests that allows clients to specify fine-grained scope claims in a JSON data structure. IUA currently does not define scopes at all and therefore does not mandate the way to convey more fine-grained scope claims in authorization requests. The committee will watch the acceptance and the development of the draft and future versions may rely on rich authorization requests.

## Closed Issues

- **Issue 1:** At this time, the method for assignment of client identifiers is not included in the profile. Registration of clients is a significant operational and security problem that is being postponed until there is more experience with problems in the field and reasonable solutions. In the field there are a variety of methods being tried. Many depend upon physical distribution methods or out of band communications to manage the authentication problems.

- **Issue 2:** This profile does support access token in JWT format and SAML 2.0 Assertions compliant with the [XUA](#) profile requirements. An operational environment must ensure, that access tokens are understood by the authorization and Resource Servers. The Bearer Token Option of earlier versions of the profile has been removed, since it does not specify the access token format and attributes used and was thus not interoperable by design.
- **Issue 3:** Audit messages are only defined for clients that are also Secure Applications. There is no defined auditing for other clients.
- **Issue 4:** This profile does not require client grouping with Secure Node or Secure Application because it is using the OAuth client authentication methods. It assumes that the client authentication method will deal with security considerations in a manner similar to the certificate management assumptions made for TLS and other certificate users.
- **Issue 8:** This profile uses the Authorization header only for conveying the authorization information. The parameter form is not prohibited but is not compliant with the profile.
- **Issue 9:** This profile does not explain the ways that some Resource Servers utilize HTTP redirects to automate some kinds of authorization activities. The actual HTTP transactions used for Obtain Access Token and Authorized RESTful Transaction are as defined within this profile. The other transactions are under the control of the Resource Server and its design.
- **Issue 10:** The selected standards are: The OAuth 2.1 Framework, JWT Token, with defined extensions, SAML Token, using the [XUA](#) extensions. |

## IHE Technical Frameworks General Introduction

The [IHE Technical Framework General Introduction](#) is shared by all of the IHE domain technical frameworks. Each technical framework volume contains links to this document where appropriate.

## 9 Copyright Licenses

IHE technical documents refer to, and make use of, a number of standards developed and published by several standards development organizations. Please refer to the IHE IHE Technical Frameworks General Introduction, Chapter 9 - [Copyright Licenses](#) for copyright license information for frequently referenced base standards.

## IHE Technical Frameworks General Introduction Appendices

The [IHE Technical Framework General Introduction Appendices](#) are components shared by all of the IHE domain technical frameworks. Each technical framework volume contains links to these documents where appropriate.

**Editor: Please update the following appendices to the General Introduction as indicated below. Note that these are not appendices to this domain's Technical Framework but rather, they are appendices to the IHE Technical Frameworks General Introduction.**

### Appendix A - Actor Summary Definitions

**Editor: Please add the following new or modified actors to the IHE Technical Frameworks General Introduction Appendix A:**

Actor	Definition
Authorization Client	A client that retrieves access tokens and presents them as part of transactions.
Authorization Server	A server that issues access tokens to requesting clients.
Resource Server	A server that provides services that need authorization.

### Appendix B - Transaction Summary Definitions

**Editor: Please add the following new or modified transactions to the IHE Technical Frameworks General Introduction Appendix B:**

Transaction	Definition
Incorporate Access Token [ITI-72]	Add an access token to a transaction.
Get Access Token [ITI-71]	Request and obtain an access token for use in authorized transactions.
Introspect Token [ITI-102]	Obtain the state and claims associated with an access token.
Get Authorization Server Metadata [ITI-103]	Obtain metadata about a Authorization Server.

### Appendix D - Glossary

**Editor: Please add the following new or modified terms to the IHE Technical Frameworks General Introduction Appendix D:**

Term	Definition
No new terms	

## Volume 1 -- Profiles

**Editor: Please add the following Section 34 to Volume 1**

The IUA Profile adds authorization information to HTTP RESTful transactions. The IUA actors and behavior can be added to other profiles and transactions that need authorization.

The actors in the IUA Profile manage the access tokens used for authorization of access to HTTP RESTful services based on the flows and transactions defined in the OAuth 2.1 Authorization Framework [OAuth 2.1]. Authorization Clients interact with the Authorization Server to retrieve access tokens and incorporate them into HTTP RESTful transactions to authorize access to resources on the Resource Servers.

The OAuth 2.1 Authorization Framework requires client identification, which may be based on a *client\_id* parameter [OAuth 2.1, Section 2.2]. Depending on the grant type, the use of the *client\_id* may be required. For example, the *Authorization Code* grant type requires the use of the *client\_id* for client identification [OAuth 2.1, Section 4.1], while the *Client Credential* grant type does not [OAuth 2.1, Section 4.2].

The OAuth 2.1 Authorization Framework also requires client authentication for confidential and credential clients [OAuth 2.1, Section 2.3]. It recommends the use of asymmetric (public-key based) methods for client authentication [OAuth 2.1, Section 9.1], but allows other suitable HTTP-based authentication schemes matching the security policy of the Authorization Server [OAuth 2.1, Section 2.3.2].

This profile requires the capability of a *client\_id* for client identification and a *client\_secret* used with the HTTP Basic Authentication scheme for client authentication of confidential and credentialed clients. Other methods for identification and authentication are allowed.

Depending on the grant type, the OAuth 2.1 Framework also requires user authentication. For example, the *Authorization Code* grant type covered by this profile requires user authentication [OAuth 2.1, Section 4.1], while the *Client Credential* grant type does not [OAuth 2.1, Section 4.2]. The methods used by the Authorization Server to authenticate the user (e.g., username and password login, session cookies, delegation to Authentication Server) are not scoped in the OAuth 2.1 Authorization Framework [OAuth 2.1, Section 3.1].

Since user authentication methods chosen depend on the projects or national security policies, they are not constrained in the IUA Profile and shall be defined in the specific implementation projects or national extensions of this profile. If the user authentication is not implemented in the Authorization Server, the use of OpenID Connect with the Authorization Grant or Hybrid flow is recommended.

It is the responsibility of the Resource Server to enforce the access policies based on the transaction performed and the information provided in the access token. Therefore, the Resource Server must be able to rely upon the decisions made by the Authorization Server (e.g., client identification, user authentication) which requires that a trust relation between the Resource Server and the Authorization Server was established beforehand.

The Resource Server may delegate the access policy enforcement to actors it is grouped with by providing the information from the access token and transaction data, or to other actors (e.g., by implementing the Authorization Decisions Verifier of the [Secure Retrieve \(SeR\)](#) Profile).

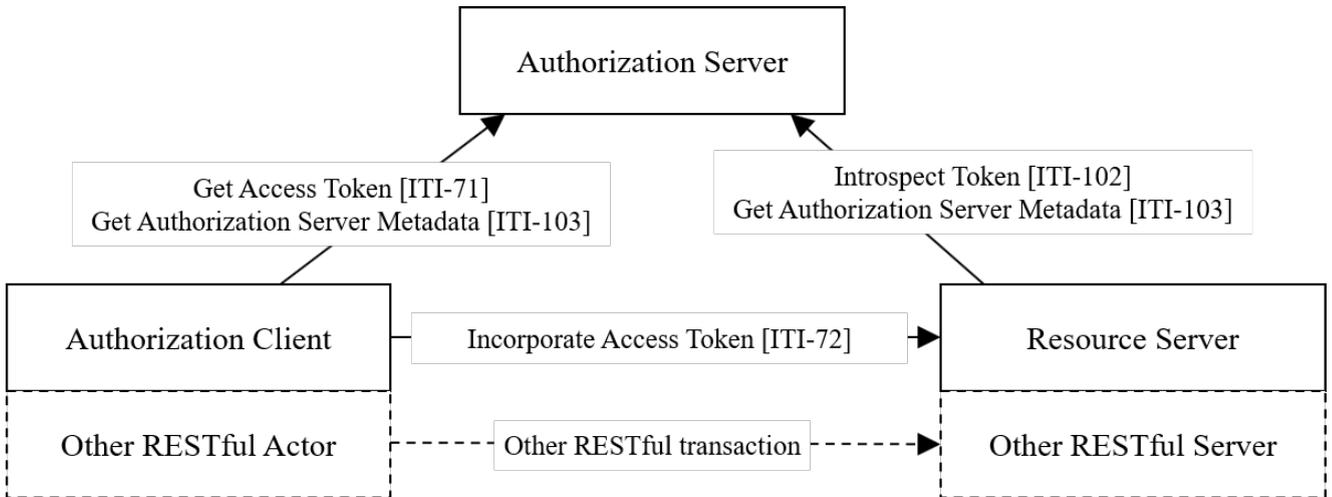
*Note:* An analogy of the segregation of duties for access control between the Authorization Server and the Resource Server is given in a textbook (*Solving Identity Management in Modern Applications*, APress 2019 by Yvonne Wilson and Abhishek Hingnikar) comparing the authorization and policy enforcement with a ticketing and entrance control to events (e.g., opera, cinema). In this analogy, the Authorization Server provides the ticket which authorizes participation, while additional checks are performed to finally participate in the event.

The IUA Profile relies on the OAuth 2.1 Authorization Framework. Since the original publication of OAuth 2.0 in 2012, several new RFCs have been published that add or remove functionality from the core specification. Version 2.1 of the OAuth Authorization Framework consolidates and simplifies the most commonly used features and best practices of OAuth 2.0.

### 34.1 IUA Actors, Transactions, and Content Modules

This section defines the actors, transactions, and/or content modules in this profile. General definitions of actors are given in the Technical Frameworks General Introduction Appendix A. IHE Transactions can be found in the Technical Frameworks General Introduction Appendix B. Both appendices are located at [http://ihe.net/Technical\\_Frameworks/#GenIntro](http://ihe.net/Technical_Frameworks/#GenIntro).

Figure 34.1-1 shows the actors directly involved in the IUA Profile and the relevant transactions between them. If needed for context, other actors that may be indirectly involved due to their participation in other related profiles are shown in dotted lines. Actors which have a required grouping are shown in conjoined boxes (see [ITI TF-1: 34.3 IUA Required Actor Groupings](#)).



**Figure 34.1-1: IUA Actor Diagram**

Table 34.1-1 lists the transactions for each actor directly involved in the IUA Profile. To claim compliance with this profile, an actor shall support all required transactions (labeled "R") and may support the optional transactions (labeled "O").

**Table 34.1-1: IUA Profile - Actors and Transactions**

Actors	Transactions	Optionality	Reference
Authorization Client	Get Access Token	R	<a href="#">ITI TF-2: 3.71</a>
	Incorporate Access Token	R	<a href="#">ITI TF-2: 3.72</a>
	Get Authorization Server Metadata	O	<a href="#">ITI TF-2: 3.103</a>
Authorization Server	Get Access Token	R	<a href="#">ITI TF-2: 3.71</a>
	Introspect Token	O (Note 1)	<a href="#">ITI TF-2: 3.102</a>
	Get Authorization Server Metadata	O	<a href="#">ITI TF-2: 3.103</a>
Resource Server	Incorporate Access Token	R	<a href="#">ITI TF-2: 3.72</a>
	Introspect Token	O (Note 1)	<a href="#">ITI TF-2: 3.102</a>
	Get Authorization Server Metadata	O	<a href="#">ITI TF-2: 3.103</a>

*Note 1:* Optionality of this transaction is "R" for an actor that supports the Token Introspection Option.

#### 34.1.1 Actor Descriptions and Actor Profile Requirements

Most requirements are documented in ITI TF-2 transactions. This section documents any additional requirements on this profile's actors.

##### 34.1.1.1 Authorization Client

The Authorization Client performs the network transactions and user interactions needed to obtain an access token and to attach that token to transactions to indicate that the transactions are authorized.

When Incorporate Access Token [ITI-72] is used with a FHIR server, an Authorization Client should query the [capabilities](#) endpoint on the Resource Server to determine if the Resource Server supports IUA. The Resource Server indicates IUA support with the element [CapabilityStatement.rest.security.service](#) and the code `iua` at system canonical URL <http://profiles.ihe.net/fhir/ihe.securityTypes/CodeSystem/securityTypes>. The CodeSystem can be retrieved from the IHE FHIR GitHub repository, [codesystem-IHE\\_securitytypes.xml](#).

Note that the Authorization Client may choose to publish a CapabilityStatement, if a CapabilityStatement is provided then the `iua` code should also be indicated to show the capability of the Authorization Client.

The Get Access Token [ITI-71] transaction is scoped to the *Authorization Code* and *Client Credential* grant types (see [ITI TF-1: 34.4.1.1 Authorization Grant Types](#)). The Authorization Client shall support at least one of the following grant types:

- *Authorization Code*

- *Client Credential*

The Authorization Client may support other grant types (see [ITI TF-1: 34.4.1.1 Authorization Grant Types](#)).

### 34.1.1.2 Authorization Server

The Authorization Server provides access tokens to requesting clients. In IUA, the Authorization Server uses an authenticated user identity, the requested HTTP RESTful service URL to the Resource Server, and/or other information to determine whether HTTP RESTful transactions are authorized. If authorized, the Authorization Server provides an access token which authorizes the client to retrieve data and documents from the Resource Server.

The Get Access Token [ITI-71] transaction is scoped to the *Authorization Code* and *Client Credential* grant types (see [ITI TF-1: 34.4.1.1 Authorization Grant Types](#)). The Authorization Server shall support the following grant types:

- *Authorization Code*
- *Client Credential*

The Authorization Server may support other grant types (see [ITI TF-1: 34.4.1.1 Authorization Grant Types](#)).

### 34.1.1.3 Resource Server

The Resource Server provides services to access protected resources that need authorization. In IUA, the Resource Server accepts an HTTP RESTful transaction request with an incorporated access token. It evaluates the access token to verify that the Authorization Server has authorized the transaction. The Resource Server shall enforce this authorization and may perform additional authorization decisions that are specific to the requested service. The Resource Server may then allow the transaction to proceed, subject to access control constraints that may be in place.

In general, Resource Servers perform additional access control decisions and may restrict responses even for transactions authorized by the Authorization Server.

When the Incorporate Access Token [ITI-72] transaction is used with a FHIR server, the Resource Server shall declare support for IUA in the [capabilities](#) endpoint using the element [CapabilityStatement.rest.security.service](#) and the code `IUA` at system canonical URL `http://profiles.ihe.net/fhir/ihe.securityTypes/CodeSystem/securityTypes`. The CodeSystem can be retrieved from the IHE FHIR GitHub repository, [codesystem-IHE\\_securitytypes.xml](#).

The Resource Servers and Authorization Servers may be grouped into an integrated product together with user authentication, access control, and other services.

## 34.2 IUA Actor Options

Options that may be selected for each actor in this profile are listed in the Table 34.2-1. Dependencies between options, when applicable, are specified in notes.

**Table 34.2-1: IUA - Actors and Options**

IUA Actor	Option	Reference
Authorization Client	Authorization Server Metadata	Section 34.2.1
Authorization Server	JWT Token (Note 1)	Section 34.2.2
	SAML Token (Note 1)	Section 34.2.3
	Token Introspection (Note 1)	Section 34.2.4
	Authorization Server Metadata	Section 34.2.1
Resource Server	JWT Token (Note 1)	Section 34.2.2
	SAML Token (Note 1)	Section 34.2.3
	Token Introspection (Note 1)	Section 34.2.4
	Authorization Server Metadata	Section 34.2.1

*Note 1:* Authorization Server or Resource Server Actors shall support at least one of the following options: JWT Token, SAML, or Token Introspection. The Authorization Client is not and does not need to be aware of the token processing options.

### 34.2.1 Authorization Server Metadata Option

This option enables automated configuration of Resource Servers and Authorization Clients by enabling them to pull configuration metadata directly from the Authorization Server.

An Authorization Client that supports this option shall have the means to be configured to interact with an Authorization Server metadata endpoint to retrieve configuration information (see [ITI TF-2: 3.103.4.2.3](#)).

Authorization Servers supporting the Authorization Server Metadata Option shall provide a metadata endpoint that provides configuration information to Authorization Client and Resource Servers. This information includes endpoint locations, supported authentication grants and signing key materials (see [ITI TF-2: 3.103.4.2.3](#)).

Resource Servers supporting the Authorization Server Metadata Option shall have the means to be configured to interact with an Authorization Server metadata endpoint to retrieve configuration information (see [ITI TF-2: 3.103.4.2.3](#)).

See [ITI TF-2: 3.103](#) for transaction requirements.

### 34.2.2 JWT Token Option

This option uses JSON Web Token encoding as access token issued by the Authorization Server. The JSON Web Token constraints are defined in [ITI TF-2: 3.71.4.2.2.1](#).

An Authorization Server that supports this option shall provide an endpoint to retrieve JWT access tokens to be incorporated in RESTful requests to Resource Servers (see [ITI TF 2: 3.71.4.1.3](#)).

A Resource Server that supports this option shall be able to accept JWT access tokens in RESTful requests, to extract and validate the information provided in the JWT access token, and to enforce access policies based on it (see [ITI TF-2: 3.72.4.3](#)).

See [ITI TF-2: 3.71.4.2.2.1](#) for transaction requirements.

### 34.2.3 SAML Token Option

This option enables integration of environments that use both the [XUA](#)-compatible SAML access token and the OAuth authorization infrastructure.

An Authorization Server that supports this option shall provide an endpoint to retrieve [XUA](#)-compliant SAML 2.0 access tokens to be incorporated in RESTful requests to Resource Servers (see [ITI TF-2: 3.71.4.1.3](#)).

A Resource Server that supports this option shall be able to accept [XUA](#)-compliant access tokens in RESTful requests, to extract and validate the information provided in the token, and to enforce access policies based on it (see [ITI TF-2: 3.72.4.3](#)).

See [ITI TF-2: 3.71.4.2.2.2](#) for transaction requirements.

### 34.2.4 Token Introspection Option

Actors that support this option use the [Token Introspection \[ITI-102\]](#) transaction to validate and obtain the claims associated with an access token. Introspection allows for custom token formats and re-evaluation of tokens after commission (enabling features like token revocation), at the cost of introspection calls to the Authorization Server.

An Authorization Server that supports the Token Introspection option shall provide an endpoint that Resource Servers will use to validate and evaluate the access token. With this option, the Authorization Server may provide token formats other than JWT or SAML. Using this option, the token can be treated as an opaque data construct for Resource Servers. When Token Introspection option is used, the Authorization Server shall provide the [Token Introspection \[ITI-102\]](#) transaction service.

A Resource Server that supports the Token Introspection Option shall have the ability to use the [Token Introspection \[ITI-102\]](#) transaction to obtain the details of the token from the Authorization Server. This transaction can be used by the Resource Server when it is uncertain about the format of the token (JWT, SAML, or opaque), or when it wants to re-evaluate the authorization policy. To be able to invoke the introspection endpoint, the Resource Server must authenticate itself. It may obtain an access token of its own from the Authorization Server for this purpose.

See [ITI TF-2: 3.102 Introspect Token](#) for transaction requirements.

## 34.3 IUA Required Actor Groupings

An actor from this profile shall implement the required transactions and/or content modules in this profile **in addition to** all of the transactions required for the grouped actor (Column 2).

**Table 34.3-1: Required Actor Groupings**

IUA Actor	Actor to be grouped with	Reference	Content Bindings Reference
Authorization Server	CT / Time Client	<a href="#">ITI TF-1:7 Consistent Time</a>	---
Resource Server	CT / Time Client	<a href="#">ITI TF-1:7 Consistent Time</a>	---
Authorization Client	---	---	---

The IUA actors are expected to be grouped with other actors that perform HTTP RESTful transaction:

- Grouping an Authorization Client with another actor means that this other actor will provide an access token as part of the HTTP transaction to a HTTP RESTful server. It may perform the Get Access Token transaction to obtain the access token.
- The corresponding HTTP RESTful server should be grouped with the Resource Server to indicate that the server can perform access control.

## 34.4 IUA Overview

### 34.4.1 Concepts

The term "authorization" and "access control" are used colloquially for a variety of related activities. All of the concepts listed below are sometimes called "authorization" or "access control". See the IHE [ITI Access Control](#) white paper for a detailed discussion of Access Control concepts. This profile will use more specific terms for each of these activities. These are:

- Provisioning -- Setting up the initial rules and updating them when the situation changes. The administrator may say "Authorize Dr. X to have access". The steps taken to make this happen are called provisioning.
- Delegation -- Adding, transferring, and revoking authorization from one person to another. This is closely related to provisioning. It differs in that it can only transfer authority that has already been provisioned, and it may track changes to provisioned access for the original person.
- Authentication -- Determining that the actual user (at the moment of authentication) is the claimed identity.
- Authorization -- Determining that the authenticated user is authorized to have access to a resource (at the moment of authorization). The profile describes how to convey authorization decisions. It is not defining how the decision is made.
- Access Control -- A system of provisioning, delegation, authentication, and authorization. It is normal to have multiple nested levels of access control. This profile is concerned with whether access is allowed to make the HTTP transaction requests to the specified resources. There are likely also building access controls, Resource Server access controls, and other access controls involved.

#### 34.4.1.1 Authorization Grant Types

The OAuth 2.1 Authorization Framework [OAuth 2.1, Section 4] defines the following authorization grant types:

- *Authorization Code Grant* [OAuth 2.1, Section 4.1]. This grant type is optimized for confidential clients who make use of User Agents (e.g., web browsers).
- *Client Credentials Grant* [OAuth 2.1, Section 4.2]. This grant type is optimized for clients requesting access tokens using only its client credentials and is restricted to confidential clients (e.g., medical devices, back end applications).

The OAuth 2.1 Authorization Framework further defines extension points to implement extension grant types [OAuth 2.1, Section 4.3], specifically:

- *Device Authorization* [RFC8628]: This grant type is optimized for devices that cannot use a browser to perform a user-agent-based authorization or don't provide interfaces for the user to input text required for authorization and authentication (e.g., medical devices, mobile health sensors).
- *SAML Client or User Assertions* [RFC7522]: To use a SAML Assertion as a credential for client or user authentication, the Authorization Client shall perform the access token request in conformance with the *SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*. This type of grant may be used to exchange a [XUA](#) Compliant SAML 2.0 assertion to a token format preferred by the Authorization Server.
- *JSON Web Token (JWT) grants* [RFC7523]: To use a JWT Token as credential for client or user authentication, the Authorization Client shall perform the access token request in conformance with the *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants*. This type of grant provides better security properties than the client credential flow as it allows for asymmetric keys to be used for client and user authentication.

The SAML and JWT grants from RFC7522 and RFC7523 may be used in federation scenarios where an Authorization Server will accept tokens from another Authorization Server as credentials for issuing access tokens. Both the RFC7522 and RFC7523 require the existence of a public key infrastructure. The operational aspects of such infrastructure are beyond the scope of this specification.

This profile specifies the use of the Authorization Code and Client Credential grant types. Actors of this profile may support other grant types as well.

#### 34.4.1.2 Terminology

The IUA Profile uses the terms "access token", "refresh token", "bearer token", "Authorization Server", "Resource Server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", and "access token response" as defined by The OAuth 2.1 Authorization Framework [OAuth 2.1].

This profile also uses the terms "token introspection", "introspection endpoint", "introspection request", and "introspection response" as defined by the OAuth2 Token Introspection specification [RFC7662].

In accordance with the definitions in the OAuth 2.1 Authorization Framework [OAuth 2.1], this profile distinguishes confidential clients, credentialed clients, and public clients as follows:

- *confidential client* - a client which stores the client authentication data (e.g., client\_id and client\_secret) in a way that the user has no access to it (e.g., a server hosted web application)
- *credentialed client* - a client which has credentials, but their identity has been not been confirmed by the Authorization Server
- *public client* - a client where the user (in principle) has access to the client code and client data. Public clients cannot store client authentication data in a confidential way (e.g., single page web applications, native mobile apps on a device, if no additional features are implemented to make the client authentication data unavailable for the user)

Note: A public client classification does not automatically mean the client is insecure. Public clients typically are under the full control of the user (e.g., a native app on the user's device) and secured against malicious attacks. Public clients just cannot hide authentication data from the user, making client authentication useless.

### 34.4.2 Use Cases

#### 34.4.2.1 Authorization Use Cases

The IUA Profile applies to use cases where a user authorizes a client application to access health data on behalf of the user using OAuth, including:

- A healthcare professional uses a server hosted web application to access a patient's electronic health record (EHR) using RESTful transactions. The web application has been registered beforehand at the Authorization Server with client identifier and client authentication method (e.g., client secret) by a system administrator. When accessing a view on the EHR, the browser is redirected to the EHR's Authorization Server. The Authorization Server authenticates the healthcare professional either by enabling the user to enter the authentication factors (e.g., username, password and 2nd factor) or by redirecting to an Identity Provider (IdP). This step may be omitted if the browser already has an authenticated session (e.g., session cookie). After user authentication, the Authorization Server performs the necessary steps to authorize the web application access to the EHR data with application specific scope, either by contract or by explicit consent given by the healthcare professional. When authorized, the web application retrieves an access token which authorizes it to request and retrieve, on behalf of the healthcare professional the EHR data from the Resource Server(s) with an application-specific scope.
- A healthcare professional uses a single page web application to access a patient's electronic health record (EHR) using RESTful transactions. At initial startup, the app registers with the Authorization Server using a dynamic client registration protocol. When accessing a view on the EHR, the steps follow the same steps as in the web application use case above.
- A clinical monitor managed by the hospital system administrators access a patients EHR using RESTful transactions in the hospital LAN. At installation time, clinical monitor has been registered at the Authorization Server with client identifier and client authentication method (e.g., client secret) by the system administrator and contracts (policies) have been deposited at the Authorization Server and Resource Server(s) authorizing access to the EHR with application specific scope. Before accessing patient EHR data, the clinical monitor requests an access token from the Authorization Server (e.g., using the client credential grant type). The clinical monitor incorporates the access token to the RESTful transactions to access EHR data and documents stored in the EHR Resource Server(s) with an application specific scope.
- A patient uses a native app on her mobile device to access data from her electronic health record (EHR) using RESTful transactions via the Internet. At first startup, the app registers with the Authorization Server using a dynamic client registration protocol. When accessing a view on the EHR, the native app is redirected to the EHR Authorization Server, which authenticates the patient either by presenting the views to enter the authentication factors (e.g., username, password and 2nd factor) or by delegating to an Identity Provider (IdP). After user authentication, the Authorization Server performs the necessary steps to authorize the native app to access to the EHR data with application-specific scope by explicit user consent. When authorized, the native app retrieves an access token which authorizes the app to request and retrieve the EHR data from the Resource Server(s) with an application specific scope on behalf of the patient.

The limitation of the scope is common to all use cases and one of the key concepts of OAuth to ensure data privacy protection by user consent. While the need for restricting the scope might be less obvious for primary system applications, it is apparent for mobile or web applications that provide specific services for patients or healthcare professionals, e.g., for medication or vaccinations or other specific use cases. Therefore, in general, an Authorization Server will present a form to the user to consent to an application specific and limited scope, e.g., to specific transactions, document types or classes, or subject-specific data the application is authorized to access.

#### 34.4.2.2 Process Flow

Figure 34.4.2.2-1 shows the basis flow of information exchange common to IUA use cases.

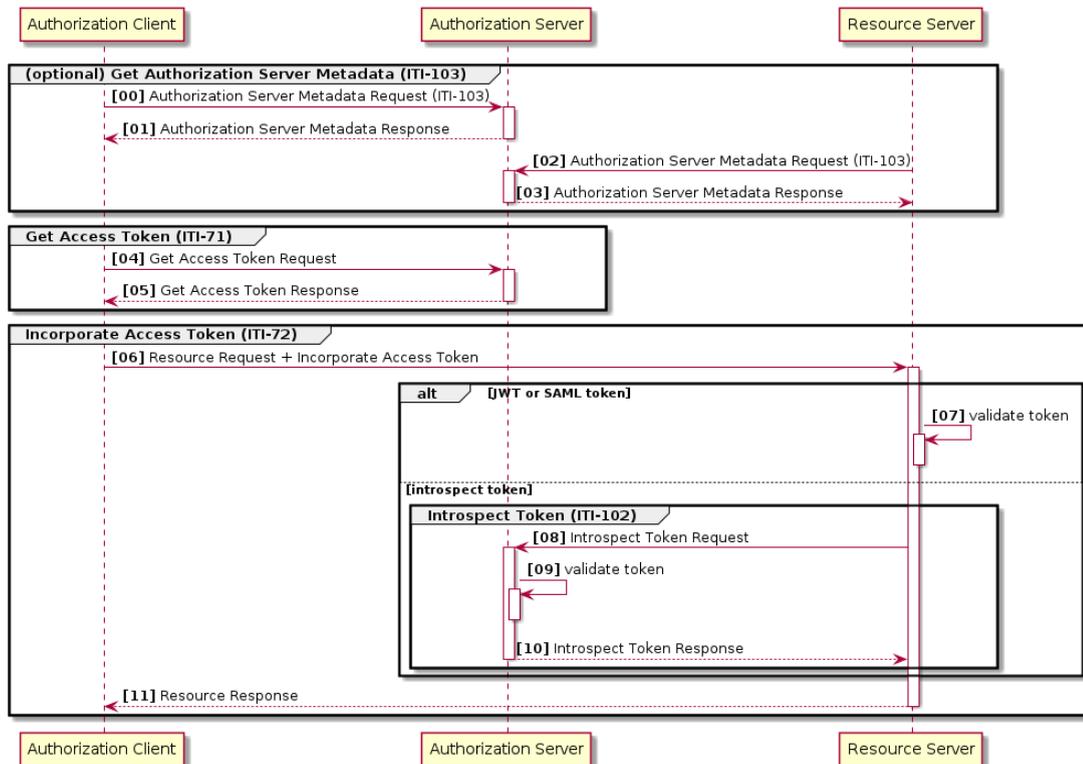


Figure 34.4.2.2-1: Basic Process of the Access Token Request and Incorporate Access Token Transaction

@startuml basic-flow

```

participant "Authorization Client" as Client
participant "Authorization Server" as AS
participant "Resource Server" as RS
  
```

```

autonumber 0 1 "<b>[00]</b>"
autoactivate on

group (optional) Get Authorization Server Metadata (ITI-103)

    Client-> AS : Authorization Server Metadata Request (ITI-103)
    AS --> Client: Authorization Server Metadata Response

    RS-> AS : Authorization Server Metadata Request (ITI-103)
    AS --> RS: Authorization Server Metadata Response

end

group Get Access Token (ITI-71)
    Client -> AS: Get Access Token Request
    Client <-- AS: Get Access Token Response
end

group Incorporate Access Token (ITI-72)

Client -> RS : Resource Request + Incorporate Access Token

alt JWT or SAML token
    RS -> RS: validate token
    deactivate RS
else introspect token

    group Introspect Token (ITI-102)
        RS -> AS : Introspect Token Request
        AS -> AS : validate token
        deactivate AS
        AS --> RS: Introspect Token Response
    end
end

Client <-- RS : Resource Response
end

@enduml

```

### Process Flow:

1. (Optional) The Authorization Client fetches the metadata document from the Authorization Server to detect the relevant authorization endpoints, and supported grant types.
2. (Optional) The Resource Server fetches the metadata document from the Authorization Server to detect the supported token format, relevant introspection endpoints, and signature verification key material.
3. The Authorization Client authenticates to the Authorization Server and (optionally) provides claims related to the intended request to access protected resources of a Resource Server.
4. The Authorization Server authenticates the client, validates the claims against any policy rules (such as user consent) and returns an access token that authorizes the Authorization Client to request the protected resources from Resource Servers. This may entail interaction with the user to establish user authentication and/or consent.
5. The Authorization Client incorporates the access token to the requests of the protected resources to Resource Servers.
6. The Resource Server obtains the claims associated with the access token. This may involve local validation of structured tokens or validation through an additional token introspection request towards the Authorization Server.
7. The Resource Server evaluates the claims and the request against a configured access policy and determines whether the client has access to the requested resources. If permitted, the Resource Server will return the requested resources.

## 34.5 IUA Security Considerations

IUA uses OAuth 2.1 which contains references to security analyses. There are also a variety of analyses available in the public literature. The IUA Profile does not introduce new considerations to those analyses and has not identified any new issues specific to healthcare.

For confidential clients, this profile requires client authentication using `client_id` and `client_secret`, if no other reliable client authentication method is used, and the client registration with a fixed `redirect_uri`.

IUA does not address issues around issuing and revoking `client_ids`. OAuth 2.1 relies upon the `client_id` for client identification and authorization. OAuth 2.1 does not further define how `client_ids` are managed.

Usually an Authorization Server publishes web forms where system administrators and users can apply for a `client_id`. OAuth 2.1 defines a mechanism for dynamically registering clients with Authorization Servers, by defining a set of desired client metadata values to be made available to the Authorization Servers [OAuth 2.1, Section 2].

The Authorization Server will have an administratively-managed list of approved `client_ids` for accepted clients. This list will be updated as new clients are approved or existing clients are removed. An access token will not be issued for unapproved clients. This assumes that the `client_id` management will deal with these security considerations in a manner similar to the certificate management assumptions made for secure communication transactions.

The Authorization Server will typically have an administratively-managed list of approved Resource Servers. The list of Resource Servers is used in access control decisions to determine if a client has access to a Resource Server. These access control decisions take place when an access token is created, are associated with the access token, and may be re-evaluated when an access token is introspected. Introspection is initiated by a Resource Server just prior to servicing a client request. Introspection can therefore be used to signal token revocation, or to provide Resource Server-specific authorization results (such as a limited view on the authorized scopes).

## 34.6 IUA Cross Profile Considerations

The Cross-Enterprise User Assertion (XUA) Profile provides equivalent functionality as IUA, but for SOAP-based transactions. Both XUA and IUA define a transaction to incorporate a token into other IHE transactions accessing protected data. These profiles have much in common, but also some remarkable differences:

- The XUA Profile defines the Provide X-User Assertion [ITI-40] transaction to retrieve an SAML 2 assertion for authorization, but does not profile the transaction, leaving the details to national extensions and project-specific implementations. In contrast, the IUA Profile specifies the analog Get Access Token [ITI-71] for OAuth 2.1-compliant access token.
- While both IUA and XUA profiles rely on the ATNA Authenticate Node [ITI-19] transaction to provide a secure communication and server side authentication; the XUA would use the "STX: TLS 1.0 Floor with AES", "STX: TLS 1.0 Floor with BCP195", "STX: TLS 1.2 Floor using BCP195" Options or the "STX WS-Security" Option to authenticate the client (or client network node); and IUA uses the "STX: HTTPS IUA" Option to leverage IUA to authenticate client applications.
- While XUA supports SAML 2 Assertions only, the IUA Profile supports access token in JWT, SAML 2.0 Assertion or custom formats (in case the Token Introspection Option is implemented).

The SAML Token Option in IUA enables an Authorization Client to incorporate access token originally retrieved and issued from a [XUA](#) X-Assertion Provider and use it in the Incorporate Access Token [ITI-72] transaction, when accessing protected data from a Resource Server via RESTful transactions.

## Volume 2 -- Transactions

**Editor: Please add Section 3.71**

### 3.71 Get Access Token [ITI-71]

#### 3.71.1 Scope

This transaction is used by an Authorization Client to retrieve an OAuth 2.1-compliant access token.

#### 3.71.2 Actor Roles

This transaction defines the following actors and roles:

**Table 3.71.2-1: Actor Roles**

Actor	Role
Authorization Client	Client requesting an access token to authorize RESTful transactions
Authorization Server	Server that grants access tokens.

#### 3.71.3 Referenced Standards

This transaction relies on standards defined in the following documents and the references therein:

- *OAuth 2.1*: The OAuth 2.1 Authorization Framework, published as draft-ietf-oauth-v2-1-01, 1 February 2021.
- *JWT Access Token*: JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens, published as draft-ietf-oauth-access-token-jwt-10, September 2020.
- *RFC7519*: JSON Web Token (JWT), May 2015.
- *RFC7522*: Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants, May 2015.
- *RFC7515*: JSON Web Signature (JWS), May 2015.
- *RFC7518*: JSON Web Algorithms (JWA), May 2015.
- *RFC4648*: The Base16, Base32, and Base64 Data Encodings, October 2006

#### 3.71.4 Messages

##### 3.71.4.1 Get Access Token Request

###### 3.71.4.1.1 Trigger Events

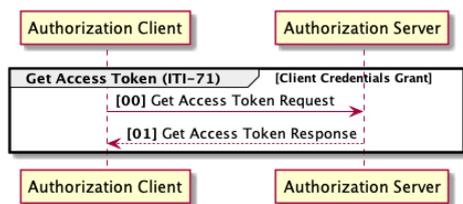
The Get Access Token Request is performed by an Authorization Client to obtain an access token to be used in further communication. The sequence of HTTP(S) requests to perform a Get Access Token transaction depends on the grant type (type of credentials) chosen.

###### 3.71.4.1.2 Message Semantics

This transaction supports two distinct grants for obtaining an access token. These grants require their own variation in message sequences as depicted in the following sections:

- Client Credential grant type - see [ITI TF-2: 3.71.4.1.2.1](#) and [3.71.4.1.3.1](#)
- Authorization Code grant type - see [ITI TF-2: 3.71.4.1.2.2](#) and [3.71.4.1.3.2](#)

###### 3.71.4.1.2.1 Client Credential grant type



**Figure 3.71.4.1.2.1-1: Sequence of HTTP requests in the client credential grant type.**

@startuml client-credential-grant

```

participant "Authorization Client" as Client
participant "Authorization Server" as AuthzServer

autonumber 0 1 "<b>[00]</b>"

group Get Access Token (ITI-71) [Client Credentials Grant]
Client ->>AuthzServer: Get Access Token Request
AuthzServer -->>Client: Get Access Token Response
end

autonumber stop

@enduml
  
```

The Authorization Client requests an access token using client credentials (or other supported means of authentication). This grant type shall be used by confidential clients only [OAuth 2.1, Section 4.2].

The Authorization Client makes a HTTP(S) POST request to the token endpoint with the following parameters in the HTTP request entity-body [OAuth 2.1, Section 4.2.2]:

- *grant\_type* (required): The value of the parameter shall be *client\_credentials*.
- *resource* (optional): Single valued identifier of the Resource Server api endpoint to be accessed [JWT Access Token, Section 3].
- *scope* (optional): The scope claimed by the Authorization Client.

The request shall use the *application/x-www-form-urlencoded* format with a character encoding of UTF-8 [OAuth 2.1, Section 4.2.2].

The Authorization Client shall present its *client\_id* and *client\_secret* in a HTTP Basic Authentication Header to the Authorization Server.

Figure 3.71.4.1.2.1-2 is a non-normative example of the access token request with client authentication using the *client\_id* and *client\_secret* in the HTTP Authorization header:

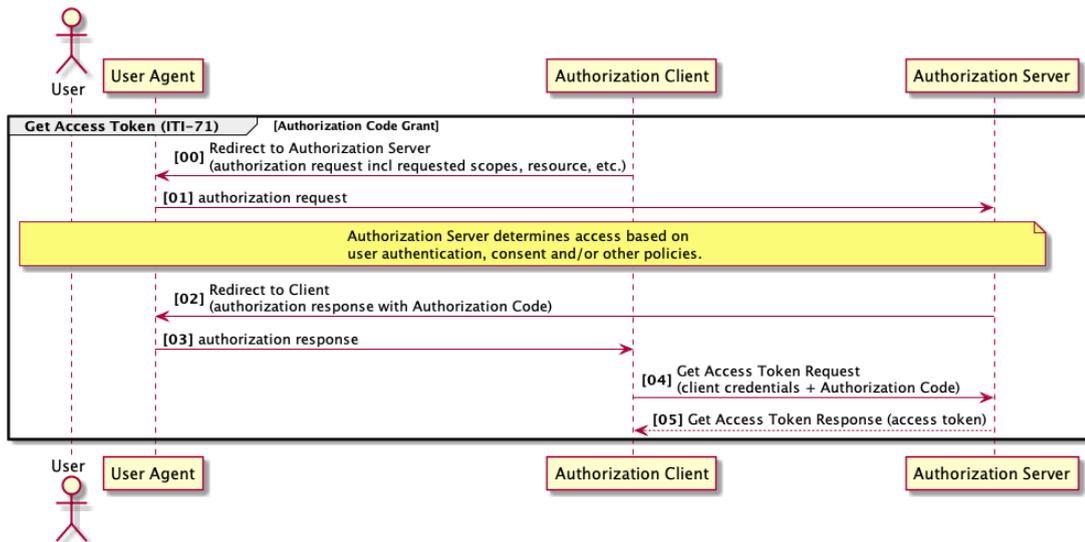
```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnwDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&scope=scope_1 scope_2 ... scope_N
&resource=https://rs.example.com/
```

**Figure 3.71.4.1.2.1-2: Example Access Token Request**

An Authorization Client that supports the Get Authorization Server Metadata Option shall use the *token\_endpoint* URL from the Authorization Server Metadata Document to obtain the access token (see [ITI TF-2: 3.103 Get Authorization Server Metadata](#)).

### 3.71.4.1.2.2 Authorization Code grant type



**Figure 3.71.4.1.2.2-1: Sequence of HTTP requests in the client authorization code grant type**

```
@startuml
    actor User
    participant "User Agent" as UserAgent
    participant "Authorization Client" as Client
    participant "Authorization Server" as AuthzServer

    autonumber 0 1 "<b>[00]</b>"

    group Get Access Token (ITI-71) [Authorization Code Grant]
        Client -> UserAgent: Redirect to Authorization Server \n(authorization request incl requested scopes, resource, etc.)
        UserAgent -> AuthzServer: authorization request

        note over User, AuthzServer
            Authorization Server determines access based on
            user authentication, consent and/or other policies.
        end note

        AuthzServer -> UserAgent: Redirect to Client \n(authorization response with Authorization Code)
        UserAgent -> Client: authorization response

        Client -> AuthzServer: Get Access Token Request \n(client credentials + Authorization Code)
        AuthzServer --> Client: Get Access Token Response (access token)
    end

@enduml
```

This grant type shall be used by confidential, credential and public clients, if the explicit consent of the user is required to authorize the Authorization Client to access data on behalf of the user.

An Authorization Client that supports the Authorization Server Metadata Option shall use the "authorization\_endpoint" URL from the Authorization Server Metadata Document to redirect the User-Agent.

The Authorization Client directs the user-agent to make a HTTP GET request to the authorization endpoint with the following parameters using the "application/x-www-form-urlencoded" format [OAuth 2.1, Section 4.1.1.3]:

- *response\_type* (required): The value must be *code*.
- *client\_id* (required, if known): The identifier the Authorization Client is registered at the Authorization Server.
- *state* (required): An unguessable value used by the client to track the state between the authorization request and the callback to the redirect URI. While this parameter is optional in the OAuth 2.1 Authorization Framework [OAuth 2.1, Section 4.1.1.3] it is required in this profile for security reasons.
- *resource* (optional): Single valued identifier of the Resource Server endpoint to be accessed [JWT Access Token, Section 3].
- *code\_challenge* (required): A challenge derived from the client generated code verifier used to correlate the authorization request to the token request [OAuth 2.1, Section 4.1.1 and references therein].
- *code\_challenge\_method* (optional): Indicator used for the code challenge transformation method. Its value may be *S256* [OAuth 2.1, Section 9.8].
- *redirect\_uri* (optional): The absolute URI of the Authorization Client callback endpoint to which the Authorization Server will send the user agent back once access is granted (or denied). This parameter is required if the Authorization Client is registered at the Authorization Server with multiple redirect URI, optional otherwise [OAuth 2.1, Section 3.1.2.3].

- *scope* (optional): The scope claimed by the Authorization Client.

Figure 3.71.4.1.2.2-2 is a non-normative example of the authorization request:

```
GET /authorize?
  response_type=code
  &client_id=s6BhdRkqt3
  &state=xyz
  &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
  &code_challenge=6fdkQaPm51113D5ukcAH3Mdx7_ntechYd1vi3n0hMZy
  &code_challenge_method=S256
  &resource=https%3A%2F%2Frs.example.com%2F
  &scope=scope_1%20scope_2%20scope_N
  HTTP/1.1
Host: server.example.com
```

#### Figure 3.71.4.1.2.2-2: Example Authorization Request

If the access request is granted (by the user or some other access policy), the Authorization Server issues an authorization code.

The Authorization Server shall redirect the user agent to the Authorization Client's redirect URI with the authorization response parameter in the *application/x-www-form-urlencoded* format. The response parameter shall be as follows:

- *code* (required): The authorization code generated by the Authorization Server.
- *state* (required): An unguessable value used by the client to track the state between the authorization request and the callback.

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=Splxl0BeZQQYbYS6wX5bIA&state=xyz
```

The Authorization Client shall use the *authorization code* in an access token request to retrieve an access token and token metadata from the Authorization Server.

An Authorization Client that supports the Get Authorization Server Metadata Option shall use the "token\_endpoint" URL from the Authorization Server Metadata Document to obtain the access token.

The Authorization Client makes a HTTP POST request to the token endpoint with the following parameters in the HTTP request entity-body [OAuth 2.1, Section 4.1.3]:

- *grant\_type* (required): The value shall be "authorization\_code".
- *code* (required): The authorization code received from the Authorization Server in the authorization response.
- *redirect\_uri* (required): The redirect URI of the Authorization Client callback. The value shall match the redirect URI the Authorization Client is registered at the Authorization Server and the value of the *redirect\_uri*, if presented in the authorization request.
- *client\_id* (required): The client identifier the Authorization Client is registered with at the Authorization Server, if no other authentication scheme is used for client authentication.
- *code\_verifier*: The original code verifier string. Required, if the "code\_challenge" parameter was used in the authorization request. Shall not be used otherwise.

Figure 3.71.4.1.2.2-3 shows a non-normative example of the access token request with client authentication using the *client\_id* and *client\_secret* in the HTTP Authorization header:

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=Splxl0BeZQQYbYS6wX5bIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_verifier=3641a2d12d66101249cdf7a79c000c1f8c05d2aafcf14bf146497bed
```

#### Figure 3.71.4.1.2.2-3: Example Access Token Request

##### 3.71.4.1.3 Expected Actions

The Authorization Server, upon receiving a Get Access Token Request, validates all request parameters.

The scope parameter incorporated in the token requests shall be used to restrict authorization grants to specific actions (e.g., restrict authorization to specific resources to read-only) and to convey claims, which at runtime are known to the Authorization Client only (e.g., if the user claims a breaking-the-glass access in an emergency situation). The Authorization Server may refuse token requests that mention scope values that are unknown to the Authorization Server.

The Authorization Client is recommended to provide a resource value to limit usability of the requested token to the intended Resource Server. If provided, the Authorization Server should evaluate any resource values provided as part of the token request procedure. The Authorization Server should execute policies to detect whether the client has access to the indicated resource. If the Authorization Client presented a resource value in the token request, the Authorization Server shall limit the list of Resource Server identifiers in the audience claim to only those that are essential to interact with the specified resource (typically only the Resource Server itself).

If the request is valid and all access policy criteria are met, the Authorization Server shall respond with the access token response as outlined [ITI TF-2: 3.71.4.2 Get Access Token Response](#).

If the authorization request is invalid, the Authorization Server shall react as defined in [OAuth 2.1, Section 4.1.2.1].

##### 3.71.4.1.3.1 Client Credential grant type

The Authorization Server shall authenticate the Authorization Client using its client identifier and secret as communicated in the HTTP Authorization header.

The Authorization Server shall verify the access token request as described in [OAuth 2.1, Section 4.2.2].

##### 3.71.4.1.3.2 Authorization Code grant type

The Authorization Server shall authenticate confidential and credential clients using the *client\_id* and *client\_secret*, or by other reliable client authentication method. In the latter case, the Authorization Server shall resolve the client authentication to a *client\_id* which was registered beforehand.

The Authorization Server shall verify that all required parameters of the authorization request are present and valid. If valid, the Authorization Server shall authenticate the user and obtain the user consent (by presenting the user a form to authorize specific scopes or by establishing approval via other means). When the user consent is established, the Authorization Server shall issue an authorization code to the client redirect URI conveying the authorization code and client state value.

The Authorization Client shall use the authorization code to request the access token from the Authorization Server. The Authorization Server shall verify the access token request as described in [OAuth 2.1, Section 4.1.3].

### 3.71.4.2 Get Access Token Response

#### 3.71.4.2.1 Trigger Event

The Get Access Token Response is returned by an Authorization Server upon request from an Authorization Client to obtain an access token to be used in further communication.

#### 3.71.4.2.2 Message Semantics

If the request does not match the requirements or is not understood, the Authorization Server shall respond with an error response as defined in the OAuth 2.1 Authorization Framework [OAuth 2.1, Section 5.2].

If the access token request is valid and authorized, the Authorization Server responds with the access token response message in JSON format with the following attributes [OAuth 2.1, Section 4.2.3]:

- *token\_type* (required): The value of the parameter shall be *Bearer*.
- *access\_token* (required): A JSON string containing the access token. The format of the access token string depends on the supported JWT Token, SAML Token or Introspect Token Options.
- *scope* (required): The scope granted by the Authorization Server.
- *expires\_in* (optional): This parameter may be used to inform the Authorization Client how long the access token is valid.
- *refresh\_token* (optional): A token provided by the Authorization Server which can be used by the Authorization Client to obtain new access tokens using the same authorization grant.

The access token response may contain other parameter or extensions depending on the implementation details of the Authorization Server [OAuth 2.1, Section 4.2.3].

The Authorization Server shall include the HTTP *Cache-Control* response header field with value *no-store* and the *Pragma* response header field value *no-cache* to the access token response [OAuth 2.1, Section 4.2.3].

Figure 3.71.4.2.2-1 is non-normative example of the access token response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "token_type": "Bearer",
  "access_token": "2YotnFZFEjr1zCsicMpAA",
  "scope": "scope_1 scope_2 ... scope_M",
  "expires_in": 3600,
  "refresh_token": "tGzv3J0kF0XG5Qx2TlKwIA",
  "example_parameter": "example_value"
}
```

**Figure 3.71.4.2.2-1: Example Access Token Response**

#### 3.71.4.2.2.1 JSON Web Token Option

The OAuth2.1 specification does not indicate the structure of the access token. Actors conforming to the JSON Web Token Option shall support access tokens formatted as signed JWT Tokens.

JWT token shall be signed as specified in JSON Web Signature [RFC7515]. If signed, the JWS Compact Serialization (base64 encoded, with single signature or MACed) shall be used as described in [RFC7515, Section 7.1].

Any actor may support the JWE (unsigned but encrypted) alternative of the JWT token.

Of the signature of JWT algorithms specified in the JSON Web Algorithms [RFC7518], the following algorithms shall be supported:

- *HS256*: HMAC using SHA-256 hash algorithm.
- *RS256*: RSA using SHA-256 hash algorithm.

The following algorithms should be supported:

- *ES256*: ECDSA using P-256 curve.
- SHA-256 hash algorithm ("ES256").

Other algorithms may be supported except the "NONE" that shall not be used.

*Note:* It is recommended to use asymmetric (public-key based, e.g., RS256) methods for signing access token. When asymmetric methods are used, a Resource Server does not need to store sensitive symmetric keys, making these methods more robust against malicious attacks.

In the JSON Web Token Option, the access token is defined as JSON object with the following attributes:

- *iss* (required): A single valued string identifying the instance which issued the access token (e.g., the Authorization Server) [JWT Access Token, Section 2.2].
- *sub* (required): If known, unique identifier of the user; the *client\_id* otherwise [JWT Access Token, Section 2.2].
- *client\_id* (required): identifier of the client for which the token is issued. [JWT Access Token, Section 2.2]
- *aud* (required): An array of identifier strings for the Resource Server endpoints to be accessed [JWT Access Token, Section 2.2]. In the special case when the JWT has one audience, the "aud" value may be a single case-sensitive string.
- *jti* (required): A unique identifier for the JWT access token [JWT Access Token, Section 2.2].
- *exp* (required): Expiration time of the token in Numeric Date format [JWT Access Token, Section 2.2].
- *scope* (required): The scope granted by the Authorization Server [JWT Access Token, Section 2.2.3].
- *nbf* (optional): The earliest time in Numeric Date format the token shall be accepted [JWT, Section 4.1.5].
- *iat* (required): The issuing date in Numeric Date format [JWT Access Token, Section 2.2].

The JWT access token may contain other parameters or extensions depending on the implementation details.

#### 3.71.4.2.2.1.1 JWT IUA extension

The Authorization Server and Resource Server shall support the following extensions to the JWT access token:

- *subject\_name* (optional): The user's name as String.
- *subject\_organization\_id* (optional): Unique identifier of the user's organization. Its value should be an URI, i.e., an URL or an OID in URN notation.
- *subject\_organization* (optional): Name or description of the user's organization.

- *subject\_role* (optional): Coded value indicating the user's role. If present, the value shall be formatted as FHIR Coding data type.
- *purpose\_of\_use* (optional): Purpose of use for the request. If a coded value is used, the value shall be formatted as FHIR Coding data type.
- *home\_community\_id* (optional): Home community identifier where the request originated. Its value should be an OID in URN notation.
- *national\_provider\_identifier* (optional): A unique identifier issued to health care providers by their national authority.
- *person\_id* (optional): Patient identifier, Citizen identifier, or other similar public identifier.

The above claims shall be wrapped in an "extensions" object with key 'ihe\_iua' and a JSON value object containing the claims. For example:

```
"extensions" : {
  "ihe_iua" : {
    "subject_name": "Dr. John Smith",
    "subject_organization": "Central Hospital",
    "subject_organization_id": "urn:oid:1.2.3.4",
    "other_value": "..."
  }
}
```

The claim content shall correspond to the content defined in the [XUA](#) specification (see [ITI TF-2: 3.40.4.1.2 Message Semantics](#)).

The mapping of IUA extension claims to [XUA](#)-compliant SAML 2.0 Assertion attributes is shown in Table 3.71.3.2.2.1-1 below.

**Table 3.71.4.2.2.1.1-1: JWT claims of the IUA extension and corresponding XUA Assertion attributes**

JWT Claim	XUA Attribute
subject_name	urn:oasis:names:tc:xspa:1.0:subject:subject-id
subject_organization	urn:oasis:names:tc:xspa:1.0:subject:organization
subject_organization_id	urn:oasis:names:tc:xspa:1.0:subject:organization-id
subject_role	urn:oasis:names:tc:xacml:2.0:subject:role
purpose_of_use	urn:oasis:names:tc:xspa:1.0:subject:purposeofuse
home_community_id	urn:ihe:iti:xca:2010:homeCommunityId
national_provider_identifier	urn:oasis:names:tc:xspa:1.0:subject:npi
person_id	<i>not defined</i>

### 3.71.4.2.2.1.2 JWT BPPC extension

In an environment that uses the IHE [Basic Patient Privacy Consents \(BPPC\)](#) Profile for documenting the consent, the Authorization Server and Resource Server shall support the following extension parameter:

- *patient\_id*: Patient identifier related to the Patient Privacy Policy Identifier. Its value should be the patient identifier in CX syntax or as URL.
- *doc\_id*: Patient Privacy Policy Acknowledgment Document. Its value should be an URN or URL.
- *acp*: Patient Privacy Policy Identifier.

If present, the claims shall be wrapped in an "extensions" object with key 'ihe\_bppc' and a JSON value object containing the claims, e.g.,

```
"extensions" : {
  "ihe_bppc" : {
    "patient_id": "543797436^^^&1.2.840.113619.6.197&ISO",
    "doc_id": "urn:oid:1.2.3.xxx",
    "acp": "urn:oid:1.2.3.yyyy",
    "other_value": "..."
  }
}
```

The mapping of IUA extension claims to XUA-compliant SAML 2.0 Assertion attributes is shown in Table 3.71.4.2.2.1.2-1.

**Table 3.71.4.2.2.1.2-1: JWT claims of the BPPC extension and corresponding XUA Assertion attributes**

JWT Claim	XUA Attribute
patient_id	urn:oasis:names:tc:xacml:2.0:resource:resource-id
doc_id	urn:ihe:iti:bppc:2007:docid
acp	urn:ihe:iti:xua:2012:acp

### 3.71.4.2.2.1.3 Example

The following is a non-normative example of JWT access token:

JOSE Header:

```
{
  "typ": "IUA-JWT",
  "alg": "HS256"
}
```

JWS Payload:

```
{
  "iss": "urn:vendor:sts",
  "sub": "b3ca1045-aa8b-42f9-9fd9-e0cbf5cb90a7",
  "aud": "http://ihe.connectathon.IUA/ResourceProvider-IHE-Connectathon",
  "exp": 1438251487,
  "nbf": 1438251187,
  "iat": 1438251187,
  "scope": "ITI-68",
  "extensions": {
    "ihe_iua": {
      "subject_name": "Dr. John Smith",
      "subject_organization": "Central Hospital",
      "subject_organization_id": "urn:oid:1.2.3.4",
      "home_community_id": "urn:oid:1.2.3.4.5.6.7.8",
      "person_id": "urn:uuid:1.2.3.4",
      "subject_role": [
        {
          "system": "2.16.840.1.113883.6.96",
          "code": "46255001",
          "display": "Pharmacist"
        }
      ]
    }
  }
}
```



	Field Name	Opt	Value Constraints
<b>Token</b> (AuditMessage/ ParticipantObjectIdentification)	ParticipantObjectTypeCode	M	"2" (System)
	ParticipantObjectTypeCodeRole	M	"13" (Security Resource)
	ParticipantObjectDataLifeCycle	U	not specialized
	ParticipantObjectIDTypeCode	U	not specialized
	ParticipantObjectSensitivity	U	not specialized
	ParticipantObjectID	U	not specialized
	ParticipantObjectName	U	not specialized
	ParticipantObjectQuery	M	URL requested
	ParticipantObjectDetail	U	not specialized

**Editor: Please add Section 3.72**

## 3.72 Incorporate Access Token [ITI-72]

### 3.72.1 Scope

This transaction is used to incorporate authorization information into HTTP RESTful transactions.

### 3.72.2 Actor Roles

**Table 3.72.2-1: Actor Roles**

Actor	Role
Authorization Client	Authorization component of a HTTP RESTful transaction client.
Resource Server	Authorization component of a HTTP RESTful transaction server.

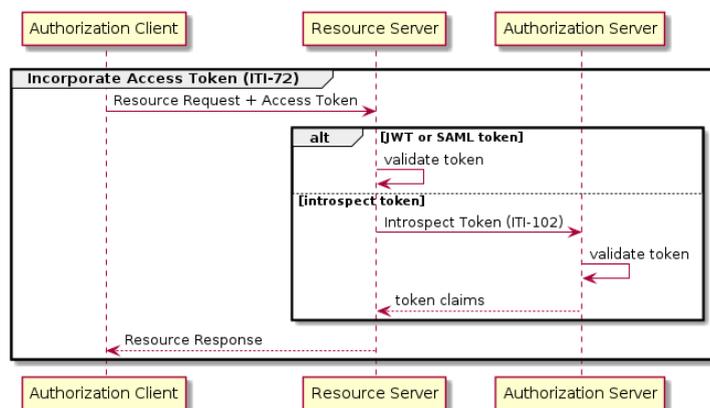
### 3.72.3 Referenced Standards

This transaction relies on standards defined in the following documents and the references therein:

- *OAuth 2.1*: The OAuth 2.1 Authorization Framework, published as draft-ietf-oauth-v2-1-01, 1 February 2021.

[XUA Cross-Enterprise User Assertion \(XUA\)](#) – Attribute Extension

### 3.72.4 Messages



**Figure 3.72.4-1: Process Flow for Incorporate Access Token Transaction**

@startuml incorporate-token

```

participant "Authorization Client" as client
participant "Resource Server" as rs
participant "Authorization Server" as as

group Incorporate Access Token (ITI-72)
client -> rs : Resource Request + Access Token

alt JWT or SAML token
rs -> rs : validate token
else introspect token
rs -> as : Introspect Token (ITI-102)
as -> as : validate token
as --> rs : token claims
end

client <- rs : Resource Response
end
@enduml

```

#### Main Flow:

1. The Authorization Client sends a resource request to a Resource Server, together with the access token. The token is opaque to the Authorization Client.
2. The Resource Server identifies the format of the token, validates the token, and obtains the claims associated with the token. To this end it may use the Authorization Server metadata, such as signing keys (for JWT or SAML tokens), or the introspect endpoint location.
3. The Resource Server makes an access control decision based upon the claims from the access token and the transaction details. It may provide access to the resource, a subset of the resource, or reject the request in total.

This transaction works in conjunction with other HTTP RESTful transaction. It extends the other transaction by adding information to the HTTP request for that HTTP RESTful transaction.

#### 3.72.4.1 Trigger Events

A client needs to make an HTTP RESTful transaction to a Resource Server that enforces access authorization.

#### 3.72.4.2 Message Semantics

The Authorization Client shall incorporate the access token as received from the Authorization Server in a HTTP Authorization header of the type *Bearer* [OAuth2.1, Bearer token].

A non-normative example of the access token incorporation to a RESTful transaction is:

```
GET /example/url/to/resource/location HTTP/1.1
Authorization: Bearer vGHtP0Jzh3QFd[...omitted for brevity...]99bhgT8YA
Host: example.com
```

#### 3.72.4.3 Expected Actions

The Resource Server determine the token format (JWT, SAML, or other) by evaluating the "access\_token\_format" value in the Authorization Server Metadata document (see [ITI TF-2: ITI-103](#)), inspection of the access token value, or some other method.

When the Resource Server is not able to process the token format (through local verification or introspection), it shall respond with HTTP 401 (Unauthorized).

The Resource Server shall validate or introspect the access token and ensure that it has not expired.

If the token includes a scope claim, the Resource Server shall verify that the scope covers the transaction to the requested resource [OAuth 2.1, Section 7].

If the token includes an audience claim, the Resource Server shall verify that the audience includes the Resource Server itself [OAuth 2.1, Section 7].

The Resource Server shall verify that the claims conveyed in the access token match the transaction type and data (e.g., verify that the patient\_id attribute of the access token corresponds to the patient\_id of the resource request).

In addition, the Resource Server shall enforce the access policies set in the specific environment by matching the information from the access token and the transaction data to the access policies.

If the token verification, or scope matching, or the access policy enforcement fails, the Resource Server shall respond with a HTTP 401 (Unauthorized) error.

Authorization Clients receiving a HTTP 401 (Unauthorized) error should NOT retry the request with the same token. An Authorization Client should reformulate the request or obtain a new token from the Authorization Server before retrying.

##### 3.72.4.3.1 JSON Web Token Option

A Resource Server that supports the JSON Web Token Option shall be able to interpret and validate the access token as a JWT Token as defined in [ITI TF-2: 3.71.4.2.2.1](#).

A Resource Server that supports the Authorization Server Metadata Option (see [ITI TF-1: 3.4.2.1](#)) shall use the keys published through the Authorization Server Metadata Document for validating the JWT token's signature.

##### 3.72.4.3.2 SAML Token Option

A Resource Server that supports the SAML Token Option shall be able to interpret and validate the access token as a [XUA](#)-compliant SAML Assertion as defined in [ITI TF-2: 3.71.4.2.2.2](#).

A Resource Server that supports the Authorization Server Metadata Option (see [ITI TF-1: 3.4.2.1](#)) shall use the keys published through the Authorization Server Metadata Document for validating the SAML token's signature.

##### 3.72.4.3.3 Introspect Token Option

A Resource Server that supports the Token Introspection Option shall use the [Introspect Token \[ITI-102\]](#) transaction to exchange the access token with the related claims (see [ITI TF-2: 3.102](#)).

A Resource Server that supports the Authorization Server Metadata Option (see [ITI TF-1: 3.4.2.1](#)) shall use the introspection endpoint published through the Authorization Server Metadata Document for validating the token and obtaining the related claims.

### 3.72.5 Security Considerations

Authorization Clients and Resource Servers that support this transaction shall fulfill the security requirements defined in the OAuth Authorization Framework [OAuth 2.1], especially:

- All HTTP transaction shall be secured by using TLS or equivalent transport security.
- Authorization Clients shall verify the identity of the Resource Server when making requests to protected resources, either by validating the TLS certificate chain [OAuth 2.1, Section 7.4.2] or by other reliable methods.

#### 3.72.5.1 Security Audit Considerations

When an [ATNA](#) Audit message needs to be generated by the Resource Server and the user is authenticated by way of a JWT Token, the [ATNA](#) Audit message **UserName** element shall record the JWT Token information using the following encoding:

**alias**"<"**user**"@"**issuer**">"

where:

- **alias** shall match the JWT token's "aud" parameter
- **user** shall match the JWT token's "sub" parameter
- **issuer** shall match the JWT token's "iss" parameter

When an [ATNA](#) Audit message needs to be generated by the Resource Server and the user is authenticated by way of a SAML Token, the [ATNA](#) Audit message **UserName** element shall record the SAML token information using the following encoding:

**alias**"<"**user**"@"**issuer**">"

where:

- **alias** shall match the optional string within the SAML Assertion's Subject element SPProvidedID attribute
- **user** shall match the required content of the SAML Assertion's Subject element
- **issuer** shall match the X-Assertion Provider entity identifier contained with the content of SAML Assertion's Issuer element

**Editor: Please add Section 3.102**

### 3.102 Introspect Token [ITI-102]

Token introspection defines a protocol that allows Resource Servers to query the Authorization Server to determine the set of claims for a given token that was presented to them by an Authorization Client. These claims include whether or not the token is currently active and the authorization context in which the token was granted.

Token introspection allows a Resource Server to query this information regardless of whether or not it is carried in the token itself (e.g., as encoded in a JWT token), allowing this method to be used along with or independently of structured token values.

The Authorization Server may decide to mark a token as active or inactive based on internal policies executed during introspection. For example, a token may be marked inactive if it has expired, been revoked, or if the introspection-requesting Resource Server is not part of the intended audience of the token. Note that the introspection response for inactive tokens does not contain other (privacy/security sensitive) claims, nor does it mention the reason why the token is marked inactive.

Additionally, token introspection is carried out in the context of a particular Resource Server, allowing the returned authorization claims to be tuned to that authorization context. For example, the Authorization Server may limit the list of scopes to those that are deemed relevant for that particular Resource Server.

#### 3.102.1 Scope

This transaction is used by Resource Servers to request an OAuth2.1-compliant bearer token evaluation from an Authorization Server.

#### 3.102.2 Actor Roles

Table 3.102.2-1: Actor Roles

Actor	Role
Resource Server	Introspection Requester
Authorization Server	Introspection Responder

#### 3.102.3 Referenced Standards

This transaction relies on standards defined in the following documents and the references therein:

- *OAuth 2.1*: The OAuth 2.1 Authorization Framework, published as draft-ietf-oauth-v2-1-01, 1 February 2021.
- *RFC7662*: OAuth 2.0 Token Introspection, published as RFC7662, October 2015

#### 3.102.4 Messages

Introspection entails a single request and response between a Resource Server and an Authorization Server. This flow is typically executed as part of the authorization of RESTful resource request as depicted in [ITITF-1: 34.4.2.2 Process Flow](#).

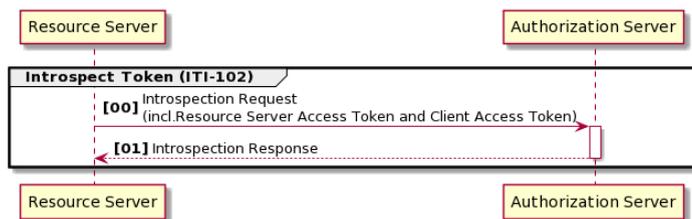


Figure 3.102.4-1: Introspect Token (ITI-102) Diagram

```
@startuml introspect
    participant "Resource Server" as ResourceServer
    participant "Authorization Server" as AuthzServer

    autonumber 0 1 "<b>[00]</b>"
    autoactivate on

    group Introspect Token (ITI-102)
        ResourceServer -> AuthzServer : Introspection Request\n(incl. Resource Server Access Token and Client Access Token)
        AuthzServer --> ResourceServer : Introspection Response
    end

    autonumber stop

@enduml
```

#### General flow:

1. Optionally, the Resource Server obtains a metadata document from the Authorization Server, providing the introspect endpoint location.
2. The Authorization Client as well as the Resource Server independently obtain their access tokens from the Authorization Server. The Resource Server requires this token to invoke the introspection endpoint.
3. The Authorization Client will invoke a Resource Server endpoint, embedding its client access token in the request.
4. The Resource Server obtains the authorization claims associated with the token from the Authorization Server by invoking the introspect endpoint. This step may be omitted if the Resource Server has a cached introspect response value for the access token (including responses where the *active* field was set to false).
5. The Resource Server will evaluate the token claims, enforce any access control policies, and return the requested resources to the Authorization Client.

#### 3.102.4.1 Token Introspection Request

This request is sent by the Resource Server to the Authorization Server to evaluate the validity and contents of an access token.

##### 3.102.4.1.1 Trigger Events

Resource Server wishes to evaluate the validity and contents of an access token.

##### 3.102.4.1.2 Message Semantics

The Resource Server shall perform an HTTP POST request to the introspect endpoint with the following parameters using the "application/x-www-form-urlencoded" format [RFC7662]:

- *token (required)*: The string value of the token. This is the Authorization Client's *access\_token* as received in the Authorization header in the request towards the Resource Server as defined in the [ITI-72] transaction.

The Resource Server must securely identify itself towards the Authorization Server by using credentials agreed between the Authorization Server and Resource

Server. See [ITI TF-2: 3.102.5 Security Considerations](#) for details.

A Resource Server that supports the Authorization Server Metadata Option (see [ITI TF-1: 3.4.2.1](#)) shall use the introspection endpoint published through the Authorization Server Metadata Document for validating the token and obtaining the related claims.

A non-normative example of an introspection request is (using a Bearer HTTP Authorization header) is:

```
POST /introspect HTTP/1.1
Host: server.example.com
Authorization: Bearer 23410913-abewfq.123483
Accept: application/json
Content-Type: application/x-www-form-urlencoded

token=2YotnFZFEjr1zCsicMpAA
```

The token being introspected may be opaque to the Resource Server. It may be formatted as JWT or SAML token, but custom formats are allowed.

#### 3.102.4.1.3 Expected Actions

Upon receiving the introspect request, the Authorization Server shall evaluate the Resource Server's access to the introspect endpoint. If access is not allowed, the Authorization Server shall return HTTP 401 (Not Authorized).

The Authorization Server shall:

- Validate the structure and authenticity of the token to be introspected.
- Validate the active state of the token (e.g., check expiry or revocation).
- Evaluate configured access policies, taking into account the authorization claims related to the token and the Resource Server identity.
- Formulate and return an introspect response.

If the one of the above checks fails the Authorization Server shall return an introspection response with the "active" field set to "false" as described in RFC7662, Section 2.2.

The Authorization Server may return different responses for the same inspected token. For example, an Authorization Server may limit which scopes from a given token are returned for each Resource Server to prevent a Resource Server from learning more about the larger network than is necessary for its operation.

The Authorization Server may include an expiry field (exp) in the introspect result. This expiry field value shall be shorter or equal to the expiry time of the access token introspected.

#### 3.102.4.2 Token Introspection Response

The Authorization Server returns the active state and authorization claims related to the token.

##### 3.102.4.2.1 Trigger Events

The Authorization Server has formulated an access policy decision for the introspected access token.

##### 3.102.4.2.2 Message Semantics

The introspect response is a JSON-formatted object. There are two variants of this object.

1. The Authorization Server considers the token to be invalid for the given Resource Server because of formatting issues, revocation data, expiry, or other access policy considerations. The introspect result object shall contain a single field "active" with boolean value "false".
2. The Authorization Server considers the token to be valid for the given Resource Server. The object shall contain a field "active" with boolean value "true". In addition, it shall contain the same fields and values as formulated for the JWT token content as specified in [ITI TF-2: 3.71.4.2.2.1 JSON Web Token](#), including defined extensions.

##### 3.102.4.2.3 Expected Actions

The Resource Server shall inspect the "active" field of the introspect result. If the active field is set to "false", the Resource Server shall return HTTP 401 (Not Authorized) for all requests carrying the introspected access token.

The Resource Server shall use the introspection results as access token claims in all access control evaluations as depicted in [ITI TF-2: 3.72.4.3](#).

The Resource Server may cache introspection results for a given access token in case the introspection result contains an expiry field. This cache shall not extend the period as defined by the expiry field. If no expiry field is provided, the Resource Server shall not cache the introspection results.

### 3.102.5 Security Considerations

The Resource Server shall securely identify itself towards the Authorization Server by using credentials agreed between the Authorization Server and Resource Server.

At minimum, the Authorization Server and Resource Server shall support the use of *Bearer* tokens for Resource Server authentication as defined in [OAuth2.1, Section 7.2]. To obtain a bearer token, the Resource Server may request such a token from the Authorization Server, reusing the client credential grant flow as described in the Get Access Token [ITI-71] transaction (see [ITI TF-2: 3.71.4.1.2.1 Client Credential Grant Type](#)).

An Authorization Server that supports the Authorization Server Metadata Option (see [ITI TF-1: 3.4.2.1](#)) should provide the available token introspection authentication options as part of its metadata document.

*Note:* The use of bearer tokens to authorize the introspect request will not lead to circular references to introspect. The Authorization Server is expected to natively understand the bearer tokens it created and therefore does not need to call introspect on itself.

Resource Servers and Authorization Servers that support this transaction shall fulfill the security requirements defined in [RFC7662], especially:

- All HTTP transactions shall be secured by using TLS or equivalent transport security.
- Resource Servers shall verify the identity of the Authorization Server, either by validating the TLS certificate chain or by other reliable methods.
- To avoid token leakage through query parameters, Resource Servers shall only use HTTP POST for the introspection request. HTTP GET is not specified, nor allowed by this transaction.
- If the token can be used only at certain Resource Servers (as indicated by the resource field in the token request), the Authorization Server shall determine whether or not the token can be used at the Resource Server making the introspection call.
- The Authorization Server shall determine whether or not the token has expired.
- If the token can be revoked after it was issued, the Authorization Server shall determine whether or not such a revocation has taken place.
- If the token has been signed (e.g., it is a JWT Token), the Authorization Server shall validate the signature.
- To avoid disclosing the internal state of the Authorization Server, an introspection response for an inactive token should not contain any additional claims beyond the required "active" claim (with its value set to "false").

#### 3.102.5.1 Security Audit Considerations

Resource Servers shall use the introspection results as authorization claims when formulating audit messages, as specified in [ITI TF-2 3.72.5.1](#).

**Editor: Please add Section 3.103**

### 3.103.1 Scope

This transaction is used by Authorization Clients and Resource Servers to obtain metadata about an Authorization Server such as endpoint locations and references to key stores required to validate (token) signatures.

### 3.103.2 Actor Roles

**Table 3.103.2-1: Actor Roles**

Actor	Role
Authorization Client	Metadata Requester
Resource Server	Metadata Requester
Authorization Server	Metadata Provider

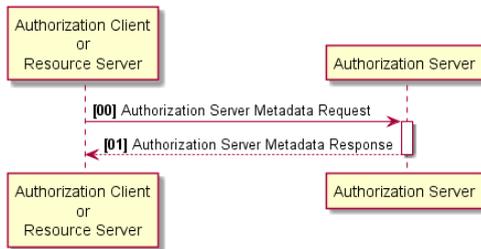
### 3.103.3 Referenced Standards

This transaction relies on standards defined in the following documents and the references therein:

- *OAuth 2.1*: The OAuth 2.1 Authorization Framework, published as draft-ietf-oauth-v2-1-01, 1 February 2021.
- *RFC8414*: OAuth 2.0 Authorization Server Metadata, June 2018
- *OpenID Connect Discovery*: OpenID Connect Discovery 1.0 incorporating errata set 1, November 2014
- *RFC7517*: JSON Web Key (JWK), May 2015

### 3.103.4 Messages

The Get Authorization Server Metadata transaction consists of a single request and response initiated by an Authorization Client or Resource Server as depicted in the diagram below:



**Figure 3.103.4-1: Get Authorization Server Metadata [ITI-103] Diagram**

```
@startuml
    participant "Authorization Client\nor\nResource Server" as Client
    participant "Authorization Server" as AuthzServer

    autonumber 0 1 "<b>[00]</b>"
    autoactivate on

    Client -> AuthzServer: Authorization Server Metadata Request
    AuthzServer --> Client: Authorization Server Metadata Response

@enduml
```

#### 3.103.4.1 Authorization Server Metadata Request

Used to request the metadata document of an Authorization Server.

##### 3.103.4.1.1 Trigger Events

An Authorization Client or Resource Server wants to learn about Authorization Server endpoint locations, key material required for signature validation, or other relevant authorization data.

##### 3.103.4.1.2 Message Semantics

The client issues a HTTP GET request to the well-known metadata endpoint associated with Authorization Server.

There are various ways of formulating this endpoint as described in [RFC8414, section 3.1], and [OpenID Connect Discovery, section 4]. Authorization Servers are recommended to follow one of these procedures, but may choose a different well-known endpoint location mechanism.

The manner in which the Authorization Clients or Resource Server are configured with the Authorization Server metadata endpoint locations is beyond the scope of this transaction.

A non-normative example of such a request using [RFC8414] is:

```
GET /.well-known/oauth-authorization-server/tenant1 HTTP/1.1
Host: example.com
```

##### 3.103.4.1.3 Expected Actions

Upon receiving a metadata request, the Authorization Server should respond with the metadata document as described in [ITI TF-2: 3.103.4.2](#).

Implementations supporting multi-tenancy shall respond with different metadata documents for different tenants, providing at least a different issuer value.

Requests for metadata for non-existing tenants shall result in a HTTP 404 (not found) response.

#### 3.103.4.2 Authorization Server Metadata Response

The response is a set of claims about the Authorization Server's configuration, including all necessary endpoints and public key information.

### 3.103.4.2.1 Trigger Events

The Authorization Server Metadata Response message is returned as response to a valid Authorization Server Metadata Request.

### 3.103.4.2.2 Message Semantics

The response is a set of claims about the Authorization Server's configuration, including all necessary endpoints and public key location information.

The response shall be structured as a plain JSON document.

The document shall be structured according to the rules set forth in [RFC8414, Section 2], with these additional constraints:

- *issuer (required)*: The Authorization Server's issuer identifier, which is a URL that uses the "https" scheme and has no query or fragment components. The URL shall use the "https" scheme. The URL shall contain the domain name of the URL at which the metadata document can be retrieved. The URL shall contain any path elements referring to a tenant. Other path elements, such as those referring to the document location (e.g., ".well-known" or "openid-configuration") should be omitted, e.g., a valid issuer associated with a metadata document retrievable at `https://example.com/.well-known/oauth-authorization-server/tenant1` would be `https://example.com/tenant1`.
- *authorization\_endpoint (required)*: Authorization Server's Authorization endpoint as used for the "authorization code" flow.
- *token\_endpoint (required)*: Authorization Server's Authorization token endpoint location.
- *jwtks\_uri (conditional)*: URL of the Authorization Server's JWK Set [RFC7517, Section 5] document. An Authorization Server that supports the JWT Token or SAML Token Options shall provide this claim to communicate the public keys that can be used to verify JWT token or SAML token signatures.
- *scopes\_supported (optional)*: The list of scopes supported by the Authorization Server. Note this transaction does not define the scopes. Authorization Servers may opt to publish a subset of the scopes usable.
- *response\_types\_supported (required)*: As the implicit grant flow is not supported in OAuth2.1, the response types should not include the value "token". Authorization Servers shall include the response type "code". Authorization Servers supporting [OpenId Connect] or other standards may include other token types, such as "id\_token".
- *grant\_types\_supported (required)*: A JSON array listing the type of grants supported toward the token endpoint. The list shall include the values "client\_credentials", and "authorization\_code".
  - Authorization Servers supporting refresh tokens shall include "refresh-token".
  - Authorization Servers supporting the JWT grant shall include "urn:ietf:params:oauth:grant-type:jwt-bearer".
  - Authorization Servers supporting the SAML grant shall include "urn:ietf:params:oauth:grant-type:saml2-bearer".
  - Authorization Servers supporting other types of grants should provide additional values to this list representing those grant types.
- *token\_endpoint\_auth\_methods\_supported (optional)*: JSON array containing a list of client authentication methods supported by this token endpoint. When provided, this list shall include "client\_secret\_basic".
- *introspection\_endpoint (conditional)*: URL of the Authorization Server's OAuth 2.0 introspection endpoint. This claim must be provided by Authorization Servers supporting the Token Introspection Option.
- *introspection\_endpoint\_auth\_methods\_supported (conditional)*: JSON array containing a list of client authentication methods supported by the introspection endpoint. This claim must be provided by Authorization Servers supporting the Token Introspection Option. As the introspect option mandates support for Resource Server bearer access tokens, this claim must include the "Bearer" value. Other authentication methods (e.g., "client\_secret\_basic") are allowed when supported by the Authorization Server.

In addition to the claims provided in [RFC8414], Authorization Servers should provide the following claim:

- *access\_token\_format (optional)*: JSON string defining the format of the access token as provided by the Authorization Server. This allows Resource Servers to learn about methods of verification. Authorization Servers supporting the JWT Token Option shall set this claim value to "ihe-jwt". Authorization Servers supporting the SAML Token Option shall set this claim value to "ihe-saml". Authorization Servers providing tokens that are non-parsable shall set this value to "opaque". Authorization Servers supporting access tokens in other structured formats may use alternative format identifiers.

### 3.103.4.2.3 Expected Actions

The Authorization Client or Resource Servers requesting the metadata document shall use the endpoint locations from the metadata document in its interactions with the Authorization Server.

The Authorization Client or Resource Servers requesting the metadata document should use the JWK values retrieved from the `jwtks_uri` endpoint published in the metadata document for validating JWT and/or SAML tokens.

Resource Servers requesting the metadata document should use the "access\_token\_format" claim to determine the method of token validation and usage (e.g., interpret and validate access token as JWT structure).

## 3.103.5 Security Considerations

All security considerations as described in [RFC8414, Section 6.2] shall be implemented.

*Note:* The issuer claim value of the metadata document is not necessarily the same as the issuer claim value associated with access-tokens. The issuer value in the metadata document refers to the source of the metadata document itself. This document may point to an token endpoint at a server other than the metadata document server. The issuer claim associated with the access token should refer to the server that implements the token endpoint.

As the metadata document is intended to support the authorization processes, there is no need to provide client credentials towards the metadata endpoint. To prevent unnecessary disclosure of credentials, the Authorization Server Metadata Request shall not include and credentials from the Authorization Client or Resource Server.

## 33 MHD Profile

**Editor: Please add the following to MHD Volume 1: Section 33.5 MHD Security Considerations**

### 33.5.1 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction specific. See the Security Considerations sections of the MHD-defined transactions for guidance on scope definition when grouped with IUA actors.

**Editor: Please add the following to MHD Volume 1: Section 33.6 MHD Cross Profile Considerations**

### 33.6.4 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction-specific. See the Security Considerations sections of the MHD-defined transactions for guidance on scope definition when grouped with IUA actors.

**Editor: Please add the following to MHD Volume 2: Section 3.65.5 Security Considerations**

### 3.65.5.2 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction-specific.

An MHD Document Source, when grouped with an IUA Authorization Client, shall use Get Access Token [ITI-71] to request the following scope from the IUA Authorization Server. This enables the MHD Document Source to submit a document using the Provide Document Bundle [ITI-65] transaction with the authorizing token in the combined transaction Incorporate Access Token [ITI-72].

The MHD Document Recipient, when grouped with an IUA Resource Server, shall require Incorporate Access Token [ITI-72] in all Provide Document Bundle [ITI-65] transactions, shall enforce the authorization decision in the token, and may further enforce policies beyond those made by the Authorization Server such as consent or business rules.

scope: ITI-65

This scope request authorizes the full ITI-65 transaction. This scope implicitly requests patient-specific Create/Update for DocumentManifest, DocumentReference, List, and Binary. Further scope refinement is allowed in realm or project-specific situations; these scopes would be in addition to the scope defined here.

**Editor: Please add the following to MHD Volume 2: Section 3.66.5 Security Considerations**

### 3.66.5.2 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction specific.

An MHD Document Consumer, when grouped with an IUA Authorization Client, shall use Get Access Token [ITI-72] to request the following scope from the IUA Authorization Server. This enables the MHD Document Consumer to use the Find Document Manifests [ITI-66] transaction with the authorizing token in the grouped transaction Incorporate Authorization Token [ITI-72].

An MHD Document Consumer, when grouped with an IUA Authorization Client, shall use Get Access Token [ITI-71] to request the following scope from the IUA Authorization Server. This enables the MHD Document Consumer to query using [ITI-66] transaction with the authorizing token in the combined transaction Incorporate Access Token [ITI-72].

The MHD Document Responder, when grouped with an IUA Resource Server, shall require Incorporate Access Token [ITI-72] in all Find Document Manifests [ITI-66] requests, shall enforce the authorization decision in the token, and may further enforce policies beyond those made by the Authorization Server such as consent or business rules.

scope: ITI-66

This scope request authorizes the full [ITI-66] transaction. This scope implicitly requests patient-specific Search/Read for DocumentManifest resources as defined in [ITI-66]. Further scope refinement is allowed in realm or project-specific situations; these additional scopes would be in addition to the scope defined here.

**Editor: Please add the following to MHD Volume 2: Section 3.67.5 Security Considerations**

### 3.67.5.2 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction specific.

An MHD Document Consumer, when grouped with an IUA Authorization Client, shall use Get Access Token [ITI-71] to request the following scope from the IUA Authorization Server. This enables the MHD Document Consumer to query using [ITI-67] transaction with the authorizing token in the combined transaction Incorporate Access Token [ITI-72].

An MHD Document Responder, when grouped with an IUA Resource Server, shall require Incorporate Access Token [ITI-72] in all Find Document References [ITI-67] requests, shall enforce the authorization decision in the token, and may further enforce policies beyond those made by the Authorization Server such as consent or business rules.

scope: ITI-67

This scope request authorizes the full [ITI-67] transaction. This scope implicitly requests patient-specific Search/Read for DocumentReference resources as defined in [ITI-67]. Further scope refinement is allowed in realm or project specific situations; these additional scopes would be in addition to the scope defined here.

**Editor: Please add the following to MHD Volume 2: Section 3.68.5 Security Considerations**

### 3.68.5.2 Use with the Internet User Authorization (IUA) Profile

The IUA Profile provides support for user authentication, app authentication, and authorization decisions. When MHD actors are grouped with IUA actors there are additional security and privacy functionality enabled by this grouping. There are additional requirements and functionality enabled through scope definitions that are transaction specific.

An MHD Document Consumer, when grouped with an IUA Authorization Client, shall use Get Access Token [ITI-71] to request the following scope from the IUA Authorization Server. This enables the MHD Document Consumer to query using [ITI-68] transaction with the authorizing token in the combined transaction Incorporate Access Token [ITI-72].

The MHD Document Responder, when grouped with an IUA Resource Server, shall require Incorporate Access Token [ITI-72] in all Retrieve Document [ITI-68] requests, shall enforce the authorization decision in the token, and may further enforce policies beyond those made by the Authorization Server such as consent or business rules.

scope: ITI-68

This scope request authorizes the full [ITI-68] transaction. This scope implicitly requests patient-specific Read for Binary resources as defined in [ITI-68]. Further scope refinement is allowed in realm or project specific situations; these additional scopes would be in addition to the scope defined here.

## 9 ATNA Profile

**Editor: Please add the following row to ATNA Volume 1: Section 9.2 ATNA Actor Options Table for both the Secure Node and Secure Application Actors**

-----|-----| "STX: HTTPS IUA" | ITI TF-1:9.2.6.7 |-----|-----|

**Editor: Please add Section 9.2.6.7 as follows**

### 9.2.6.7 STX: HTTPS IUA Option

Actors that support this option utilize server-side authenticated TLS (also known as https) to authenticate the server to the client and provide communications integrity and encryption. An actor that supports this option shall be grouped with actors in the Internet User Authorization (IUA) Profile to authenticate a client application to the server (IUA Resource Server). See [ITI TF-2: 3.19.6.6](#).

**Editor: Please add the following section to Volume 2 as follows**

### **3.19.6.6 STX: HTTPS IUA Option**

This configuration utilizes server-side authenticated TLS (also known as https) to authenticate the server to the client and provide communications integrity and encryption; and the IUA Profile to authenticate the client application to the server (IUA Resource Server).

The ATNA Local Secure Node or Secure Application shall be grouped with an IUA Authorization Client. The ATNA Remote Secure Node or Secure Application shall be grouped with an IUA Resource Server.

- TLS shall be server side authenticated, and may be client authenticated
- TLS shall be compliant with BCP195
- Local Secure Node or Secure Application shall reject connections that are not https, and may enforce other policies
- Remote Secure Node or Secure Application shall reject connections that do not carry a valid IUA token, and may enforce other policies