ADSS Server v8.4.2 –

Unity Service Developers Guide

ASCERTIA LTD

AUGUST 2026

Document Version – 1.0.0

© Ascertia Limited. All rights reserved.
This document contains commercial-in-confidence material. It must not be disclosed to any third party without the written authority of Ascertia Limited.

CONTENTS

1 Introduction 4

1.1 Scope 4
1.2 Intended Readership 4
1.3 Conventions 4
1.4 Technical support 4

2 ADSS Server Unity Service Overview 5

3 Business Application Interfaces 7

3.1 Ascertia APIs 7
3.1.1 Service Status 7
3.1.2 Get Profile Info 7
3.1.3 Register User 9
3.1.4 Update User 11
3.1.5 Delete User 12
3.1.6 Get User 13
3.1.7 Get Users 14
3.1.8 Change Password 16
3.1.9 Recover Password 17
3.1.10 Confirm Recover Password 19
3.1.11 Change User Email 20
3.1.12 Confirm Change User Email 22
3.1.13 Change User Mobile 23
3.1.14 Confirm Change User Mobile 25
3.1.15 Get Registered Devices 26
3.1.16 Delete Device 28
3.1.17 Generate Key Pair 29
3.1.18 Delete Key Pair 30
3.1.19 Get CSR 31
3.1.20 Import Certificate 33
3.1.21 Get User's Certificates 35
3.2 CSC APIs 37
3.2.1 Info 37
3.2.2 Authentication/Login 40
3.2.3 Authentication/Revoke 42
3.2.4 Credentials/List 44
3.2.5 Credentials/Info 49
3.2.6 Credentials/Authorize 54

3.2.7 Credentials/Authorizecheck 57
3.2.8 Credentials/GetChallenge 59
3.2.9 Credentials/extendTransaction 60
3.2.10 Signatures/SignHash 62
3.2.11 Signatures/SignDoc 67
3.2.12 Signatures/Timestamp 75
3.2.13 OAuth2/Authorize 76
3.2.14 OAuth2/PushedAuthorize 81
3.2.15 OAuth2/Token – Authorization Code Flow 87
3.2.16 OAuth2/Token – Client Credentials Flow 91
3.2.17 OAuth2/Token – Refresh Token Flow 92
3.2.18 OAuth2/Revoke 95

4 Mobile Application Interfaces 97

4.1 Authenticate Client 97
4.2 Authenticate User 98
4.3 Verify OTPs 101
4.4 Renew Access Token 103
4.5 Device Registration 104
4.6 List Registered Devices – User Access Token 105
4.7 List Registered Devices – Client Credentials 106
4.8 Delete Device 108
4.9 Get a Pending Authorisation Request 109
4.10 Get Pending Authorisation Requests 111
4.11 Authorise a Pending Request 115
4.12 Cancel a Pending Authorisation Request 118
4.13 Users Profile 119
4.14 Get Device Registration Settings 120
4.15 Generate QR Code 121
4.16 Verify QR Code 123
4.17 Register Device for Push Notification 124
4.18 Delete Device for Push Notification 125

5 Signature Activation Data (SAD) – Body Structure 126

6 Updates 128

7 Error Code List 129

1 Introduction

1.1 Scope

This document provides information on how to integrate mobile applications and business applications with ADSS Server UNITY Service for remote signature authorisation.

The integration uses REST architectural style APIs only. These calls are sent over HTTPS from the mobile device to the ADSS Server UNITY Service.

1.2 Intended Readership

This guide is intended for developers who are integrating mobile applications with ADSS Server for remote signature authorisation. The document assumes a reasonable knowledge of web application development, specifically RESTful Web services and ADSS Server.

1.3 Conventions

The following typographical conventions are used in this guide to help locate and identify information:

- **Bold** text identifies menu names, menu options, items you can click on the screen, file names, folder names, and keyboard keys.
- Courier New font identifies code and text that appears on the command line.
- **Bold Courier New** identifies commands that you are required to type in.
- Courier New font identifies Ajax request/response in HTTP message body.

1.4 Technical support

If technical support is required, Ascertia has a dedicated support team. Ascertia Support can be contacted in the following ways:

Support Website	Ascertia Community Portal
Support Email	support@ascertia.com
Knowledge base	Ascertia Community Portal

In addition to the free support service describe above, Ascertia provides formal support agreements with all product sales. Please contact sales@ascertia.com for more details.

A Product Support Questionnaire should be completed to provide Ascertia Support with further information about your system environment. When requesting help, it is always important to confirm:

- System Platform details.
- ADSS Server version number and build date.
- Details of specific issue and the relevant steps taken to reproduce it.
- Database version and patch level.
- Product log files

2 ADSS Server Unity Service Overview

ADSS Server Unity Service is the client-facing component of the ADSS Server remote signing solution. It acts as a gateway controlling access to the ADSS Server Signature Activation Module (SAM) which performs the actual remote signing operation. For brevity the ADSS Server Unity Service will be referred to as ADSS Unity throughout this document.

The purpose of ADSS Unity is to manage:

- **Unity registration services:**
  - Register users for remote signing. This involves not only registering the user details (e.g. name, email and phone number) but also requesting their signing key pair generation inside the ADSS Server SAM's HSM and then ensuring the corresponding public key certificate is issued by communicating with various ADSS Server components (and optionally any external CAs).
  - Register user's mobile devices for remote signing. It is possible for a user to register multiple devices.
- **Unity signing services:**
  - Receiving signing requests from business applications on behalf of users. Note that the business applications can directly interact with Unity Service.
  - Request authorisation of the remote signature from the user, by conducting a Signature Activation Protocol (SAP) with the user's registered mobile device.

Note for both registration and signing ADSS Unity Service is not the end-point, it acts as a front-end management service for the ADSS Server SAM service.

ADSS Unity has an Ascertia-defined API for user registration, device registration and certificate management and follows the industry-defined Cloud Signature Consortium[1] v2 protocol for signing operations. The Signature Activation Protocol (SAP) interface with the user's mobile device for authorising the remote signature is also Ascertia-defined.

[1] See http://www.cloudsignatureconsortium.org/ for more details

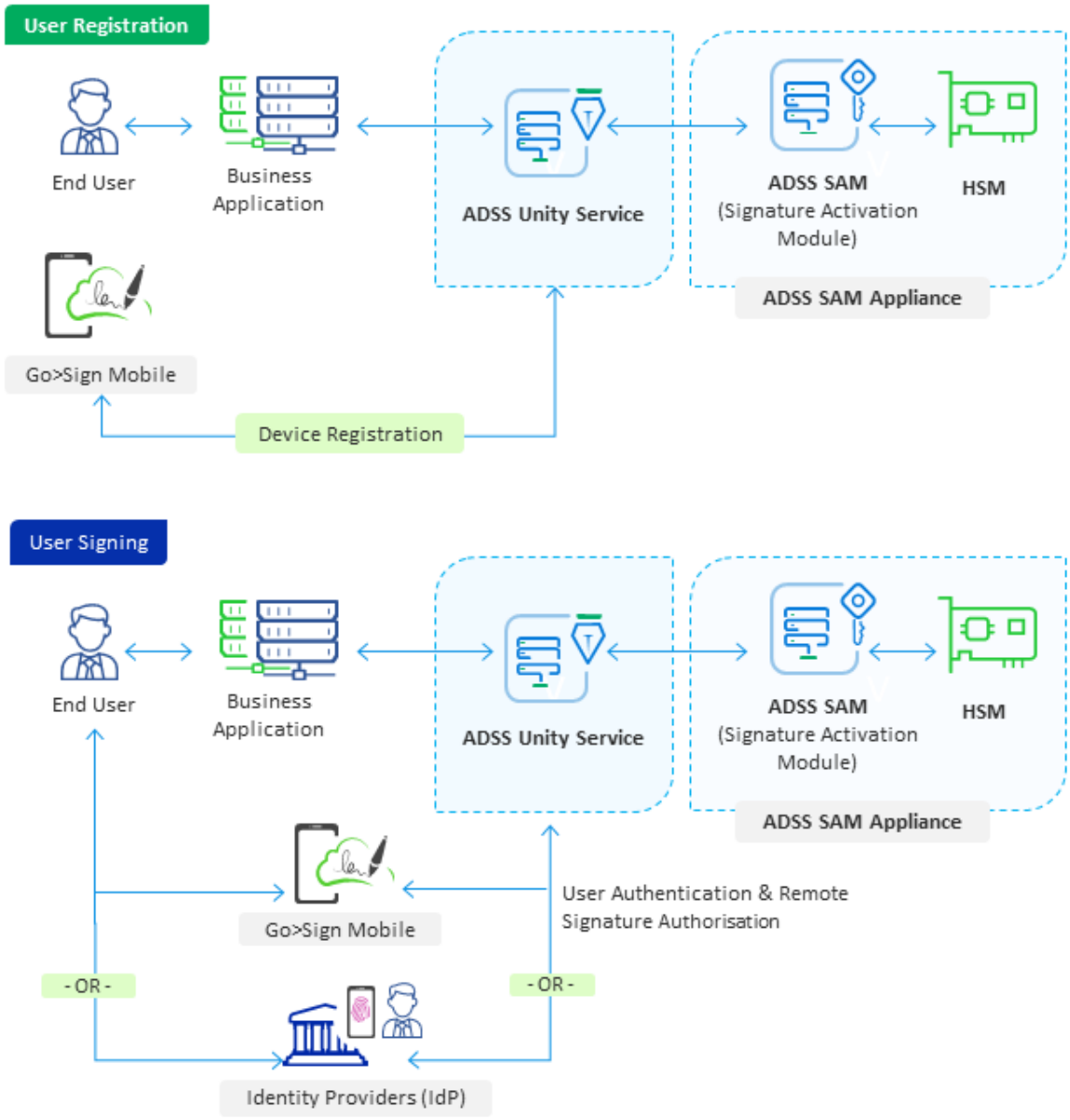

Figure 1 - UNITY Service & Business Application Interaction

ADSS Unity Service receives all the benefits of the well-proven, robust architecture of ADSS Server. The ADSS Server Architecture & Deployment Guide describes how to implement a high availability and fault tolerant solution.

Calls to ADSS Services, including the Unity Service, use standard ADSS Server Tomcat HTTPS Listeners/Connectors. Port 8778 is used to communicate with ADSS Server over server-side TLS v1.2 and TLS v1.3.

3 Business Application Interfaces

ADSS Unity Service has a number of APIs aimed at business applications which initiate user registrations and signing operations. We can categorise the APIs in two sections:

- Ascertia APIs
- CSC APIs

The details of both APIs are given below:

3.1 Ascertia APIs

The APIs implemented by Ascertia for ADSS Unity Service is given below:

3.1.1 Service Status

This API is used to get the status of Unity Service whether its running, stopped or disabled. Business applications can use this API to test the connectivity with Unity Service.

https://<server>:8778/adss/service/unity		
HTTP Method	GET	
Accept	application/json	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	{ "message": "success", "message_description": "ADSS Unity Service is running" }
400	Bad Request	
401	Unauthorised	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 1 – Service Status

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

3.1.2 Get Profile Info

This interface returns the information of a Unity Service profile e.g. all settings configured in that profile. The business application will send the profile ID and client ID in request and Unity Service will return the information of that profile in response.

Exposed for: Business Applications

https://server:8779/adss/service/unity/v1/profile/{profile_id}

HTTP Verb	GET	
Accept	application/json	
Authorization	{client_access_token | user_access_token}	
Status Code	**Message**	**Response Body**

200	OK	{
		"profile_id":"adss:unity:profile:001",
		"profile_name": "adss:unity:profile:001",
		"profile_status": "ACTIVE",
		"basic_authentication": true,
		"oauth2_authentication": true,
		"credentials_authorisation_method": "IMPLICIT",
		"authentication_with_qr_code": false,
		"no_authentication": false,
		"sam_profile": {
		"profile_id": "adss:sam:profile:001",
		"profile_name": "adss:sam:profile:001",
		"profile_status": "ACTIVE",
		"crypto_profile": "utimaco",
		"key_type": "RSA",
		"key_size": 2048,
		"kak_size": 2048,
		"signature_padding_scheme": "PKCS1",
		"compute_hash_at_signing": true,
		"hash_algorithm": "SHA256",
		"bulk_signing_allowed": false,
		"number_of_hashes": 0,
		"device_key_type": "ECDSA",
		"device_key_size": 256,
		"secure_element_required": true,
		" biometric_required ": true,
		"allowed_device": 10,
		"user_authorisation_settings": "ONCE",
		"sole_control_type": "2",
		"authorisation_request_expiry": "100",
		"authorisation_request_expiry_unit": "SECONDS" | "MINUTES" | "HOURS" | "DAYS" | "MONTHS" | "YEARS",
		"sad_expiry": "100",
		"sad_expiry_unit": "SECONDS" | "MINUTES" | "HOURS" | "DAYS" | "MONTHS" | "YEARS" | "CUSTOM_DATE",
		"clock_tolerance_on_auth_cert": "10"
		},
		"saml_asscertion": {
		"idp_signing_certificate": ""
		"identify_user_id": "SAML_ATTRIBUTE_NAME",
		"identify_user_attribute": "abc"
		},	

		"authentication_with_otp": {
		"sms_otp": true,
		"email_otp": true
		},
		
		“authorisation_certificate” : “auth_cert_alias”,
		
		"delegated_authentication": true,
		
		"external_idp": "IdP Connector"
		}

400	Bad Request	Device token is missing
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 2 – Get Profile Information

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.
profile_id	MANDATORY	*String*	Profile ID or Profile Name whose information is required in response (max. 100 characters).

3.1.3 Register User

Creates a user in SAM Service. When a new user is created then response status ‘201’ is returned. A business application will register its users using this interface.

Note: To ensure password recovery is possible, it’s recommended to set an email or mobile number for users during registration. If these details are missing, the BA can later add them using the Update User API.

https://<server>:8778/adss/service/unity/v1/users	
HTTP Verb	POST
Content-Type	application/json
Authorization	Bearer {client_access_token}
Accept	application/json
Request Body	{ "user_id": "johnDoe12", "app_name": "Application_01", "user_name": "John Doe", "user_password": "password",

	"user_email": "john.doe@ascertia.com", "user_mobile": "00448007720442", "profile_id": "profile-001" }

Status Code	Message	Response Body
201	Created	
200	OK	
400	Bad Request	
401	Unauthorised	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 3 – Register User

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.
user_id	MANDATORY	*String*	User ID identifying the registered user in Unity service (max. 50 characters and allowed characters are **a-zA-Z0-9_.@-**).
app_name	OPTIONAL	*String*	Name of the organization that requested the client's Business Application to register the user. The Business Application can be dealing with multiple organizations so it can store the actual organization's information with its users to identify users of a particular organization (max. 50 characters). The same parameter can be used later as search filter in *Get Users API*.
user_name	MANDATORY	*String*	User name as friendly name for the registered user in Unity service (max. 200 characters). Following languages are supported for username: • English Characters • Norwegian Characters • Slovenian Characters • Czech & Slovak Characters • Icelandic Characters • Arabic Characters • Latvian Characters
user_password	CONDITIONAL	*String*	Password for the registered user in Unity service. Mandatory in case where Basic Authentication check is enabled in Unity profile (max. 500 characters).
user_email	OPTIONAL	*String*	Email for the registered user in Unity service. It will be used to send OTP for mobile device registration etc. (max. 100 characters).

user_mobile	OPTIONAL	*String*	Mobile number for the registered user in Unity service. It will be used to send OTP for mobile device registration etc (max. 100 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

3.1.4 Update User

Updates a user's information. When a user is updated then response status '200' is returned. A business application will update its user's information using this interface.

https://<server>:8778/adss/service/unity/v1/users/{user_id}		
HTTP Verb	PUT	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body	{ "user_name": "John Doe", "user_email": "john.doe@ascertia.com", "user_mobile": "00448007720442", "profile_id": "profile-001", "status": "INACTIVE" }	
Status Code	**Message**	**Response Body**
201	Created	
200	OK	
400	Bad Request	
401	Unauthorised	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 4 – Update User

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.

user_id	CONDITIONAL	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
user_name	OPTIONAL	*String*	User name as friendly name for the registered user in Unity service (max. 50 characters).
user_email	OPTIONAL	*String*	Email for the registered user in Unity service (max. 100 characters).
user_mobile	OPTIONAL	*String*	Mobile number for the registered user in Unity service (max. 100 characters).
status	OPTIONAL	*String*	Status of the user in Unity Service. The status of a user can be updated using the values (ACTIVE/INACTIVE/BLOCKED).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

3.1.5 Delete User

Deletes a user in Unity Service identified by {*user_id*}. This interface will be used by a business application to remove a user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/{user_id}?profile_id=xyz		
HTTP Verb	DELETE	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 5 – Delete User

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.

user_id	CONDITIONAL	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.6 Get User

Returns a user's information registered in Unity Service identified by {*user_id*}. A business application will use this interface to get a user's information.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/{user_id}?profile_id=xyz		
HTTP Verb	GET	
Content-Type		
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	{ "user_id": "johnDoe12" "user_name": "John Doe", "app_name": "Application_01", "user_email": "john.doe@ascertia.com", "user_mobile": "00448007720442", "status": "ACTIVE", "created_at": "2020-12-15 12:19:39", "last_updated_at": "2020-12-15 12:22:19" "profile_id": "adss:sam:profile:001", }
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 6 – Get User

Request Parameters

Parameters	Presence	Value	Description

Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID identifying the registered user in UNITY service (max. 50 characters).
user_name	MANDATORY	*String*	User name as friendly name for the registered user in Unity service (max. 50 characters).
user_email	MANDATORY	*String*	Email for the registered user in Unity service (max. 100 characters).
user_mobile	MANDATORY	*String*	Mobile number for the registered user in Unity service (max. 100 characters).
status	OPTIONAL	*String*	Status of the user in Unity Service. The status of a user can be (ACTIVE/INACTIVE/BLOCKED).
created_at	MANDATORY	*String*	The date on which user is created (max. 50 characters).
profile_id	MANDATORY	*String*	It's a SAM Service Profile ID that is associated with the user (max. 50 characters).
app_name	OPTIONAL	*String*	Application name to be used by business application for listing of users (max. 50 characters).
last_updated_at	MANDATORY	*String*	The date on which user information is modified (max. 50 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.7 Get Users

Returns all the users information registered in Unity Service for a particular client identified by client access token. Paged results can also be fetched using the *start_pointer* and *fetch_size* parameters.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/{start_pointer}/{fetch_size}?{query_params}	
HTTP Verb	GET
Content-Type	
Accept	application/json
Authorization	Bearer {client_access_token}
Request Body	

Response Headers		
x-total-records	2	
Status Code	**Message**	**Response Body**

200	OK	[{
		"user_name": "John Doe",
		"user_id": "johnDoe12",
		"app_name": "Application_01",
		"user_email": "john.doe@ascertia.com",
		"user_mobile": "00448007720442",
		"status": "ACTIVE",
		"created_at": "2017-10-10 10:30:00",
		"last_updated_at": "2017-10-10 10:30:00"
		"profile_id": "adss:sam:profile:001"
		},{
		"user_name": "Peter Doe",
		"user_id": "peterDoe12",
		"app_name": "Application_01",
		"user_email": "peter.doe@ascertia.com",
		"user_mobile": "00448007720442",
		"status": "ACTIVE",
		"created_at": "2017-10-10 10:30:00",
		"last_updated_at": "2017-10-10 10:30:00"
		"profile_id": "adss:sam:profile:001"
		}]

404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 7 – Get Users

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client authentication.
{query_params}	OPTIONAL	*String*	Currently supported parameters are: • profile_id • app_name Response will contain the users that belong to a particular app_name e.g. …/service/unity/v1/keypairs/cert/list/my_client_01/user_01?client_id=abc&profile_id=xyz&app_name=application01
start_pointer	MANDATORY	*Integer*	Its the starting index of the data to be extracted.
fetch_size	MANDATORY	*Integer*	Its batch size client wants to fetch from Unity Service.

Response Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
user_name	OPTIONAL	*String*	User name as friendly name for the registered user in Unity service (max. 50 characters).
user_email	MANDATORY	*String*	Email for the registered user in Unity service (max. 100 characters).
user_mobile	MANDATORY	*String*	Mobile number for the registered user in Unity service (max. 100 characters).
status	MANDATORY	*String*	Status of the user in Unity Service. The status of a user can be: • ACTIVE • INACTIVE • BLOCKED
created_at	MANDATORY	*String*	The date on which user is created (max. 23 characters).
last_updated_at	MANDATORY	*String*	The date on which user information is modified (max. 23 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.8 Change Password

This interface is used to change the password of a user. The user provides the old password and new password in request. The Unity verifies the old password and after successful verification, it will change the old password with the new one.

Note: This interface will only be used if a password was provided at the time of user registration, otherwise it is of no use and the server will return error 'Unauthorized' as there will be no password stored against the user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/change/password		
HTTP Verb	PUT	
Content-Type	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Accept	application/json	

Request Body	{
	"user_id": "johnDoe12",
	"profile_id": "profile-001",
	"user_password": "old-password",
	"user_new_password": "new-password"
	}	

Status Code	**Message**	**Response Body**
200	OK	
400	Bad Request	

401	Unauthorized	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 8 – Change Password

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
user_password	MANDATORY	*String*	Old password of the user that he/she wants to change (max. 500 characters).
user_new_password	MANDATORY	*String*	New password of the user (max. 500 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.9 Recover Password

Initiates the user password recovery process. If a user forgets their password, this interface can be used to recover/reset it. Password recovery is done via business application calls this interface to initiate the process. Then, Unity will send the OTP(s) based on the OTP_TYPE provided in the header. The user can specify either SMS_OTP, EMAIL_OTP, or both. If OTP_TYPE is not specified, OTPs will be sent to both the user's mobile and email by default. The client will then send these OTPs in a separate call using another interface, which is discussed in the next section.

Note: if the user was registered without password, this interface can also be used to set a password for that user. Additionally, if a user was registered without an email or mobile number and needs to recover their password, begin by updating their profile with a valid email or mobile number using the Update User API. This will enable password recovery by allowing an OTP to be sent to the user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/password/recover	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json

Authorization	Bearer {client_access_token | user_access_token}	
OTP_TYPE header	This header tells the UNITY Service what type of OTP(s) to generate. It can contain one of the following values: • SMS_OTP • EMAIL_OTP • SMS_OTP, EMAIL_OTP **Note:** These values are not case sensitive and clients can send "sms_otp" or "email_otp" as well. Similarly, the name of the header OTP_TYPE is also case insensitive and otp_type will also be accepted by UNITY.	

Request Body	{
	"user_id": "johnDoe12",
	"profile_id": "profile-001"
	}	

Status Code	**Message**	**Response Body**

200	OK	If two OTPs will be sent to user:
		[
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		},
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}
		]

		If one OTP will be sent on user email:
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		}

		If one OTP will be sent to user's mobile:
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}

400	Bad Request	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 9 – Recover Password

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.

user_id	CONDITIONAL	*String*	User ID identifying the registered user in Unity service (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
type	CONDITIONAL	*String*	As OTP can be sent on both mediums i.e email/mobile so it defines the type of OTP (max. 20 characters). There can be following types: - EMAIL_OTP - SMS_OTP
sent_to	CONDITIONAL	*String*	It could be the mobile number or email of the user depends upon the "type" of OTP (max. 500 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.10 Confirm Recover Password

Completes user password recovery process. The business application will send the OTPs and new password in request using this API, and the Unity will first validate the OTP and after successful validation, change the old password with the new password.

Note: When a user is registered without an email or mobile number, the Operator will generate an OTP via the console and share it with the user. The user must then send the shared OTP in both the SMS and EMAIL types when calling the Confirm Password Recovery API.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/password/recoverconfirm		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	

Request Body	{
	"user_id": "johnDoe12",
	"profile_id": "profile-001",
	"sms_otp": "225665",
	"email_otp": "654456",
	"user_password": "P@$$w0rD!@"
	}
	
	If only one OTP will be received by user either on SMS or email, the request would contain only "sms_otp" or "email_otp".	

Status Code	**Message**	**Response Body**

200	OK	
400	Bad Request	
401	Unauthorized	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 10 – Confirm Recover Password

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
sms_otp	MANDATORY	*String*	OTP received on the user's registered mobile number (max. 100 characters).
email_otp	MANDATORY	*String*	OTP received on the user's registered email (max. 100 characters).
user_password	MANDATORY	*String*	New password for the registered user (max. 500 characters).

Response Parameters

Parameters	Presence	Value	Description
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.11 Change User Email

This interface will be used by a business application to change a user's email. The process completes in two steps: in the first step, the business application sends the user ID and new email address to this interface, and Unity will send OTP(s) based on the OTP_TYPE provided in the header. The user can specify either SMS_OTP, EMAIL_OTP, or both. If OTP_TYPE is not specified, OTPs will be sent to both the user's mobile and the new email address by default. The business application will then send these OTPs in another request using another interface, which is discussed in the next section.

Note: A user's email can also be changed using the "Update User" API but this API provides more security and control while changing the email. The clients can choose the relevant APIs to change a user's email according to their requirements and policies.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/email/change	
HTTP Verb	POST

Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
OTP_TYPE header	This header tells the UNITY Service what type of OTP(s) to generate. It can contain one of the following values: • SMS_OTP • EMAIL_OTP • SMS_OTP, EMAIL_OTP **Note:** These values are not case sensitive and clients can send "sms_otp" or "email_otp" as well. Similarly, the name of the header OTP_TYPE is also case insensitive and otp_type will also be accepted by UNITY.	

Request Body	{
	"user_id": "johnDoe12",
	"user_email": "john.doe@ascertia.com",
	"profile_id": "profile-001"
	}	

Status Code	**Message**	**Response Body**

200	OK	If two OTPs will be sent to user:
		[
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		},
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}
		]

		If one OTP will be sent on user email:
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		}

		If one OTP will be sent to user's mobile:
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}

400	Bad Request	
403	Forbidden	
404	Not Found	
500	Internal Server Error	

429	Too Many Requests	

Table 11 – Change User Email

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
user_email	MANDATORY	*String*	New Email for the registered user (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
type	MANDATORY	*String*	Type of the OTP e.g. sms/email (max. 100 characters).
sent_to	MANDATORY	*String*	Mobile number or email of the user where OTP is sent (max. 500 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.12 Confirm Change User Email

Once the OTPs are received by the user for change email, the business application will provide these OTPs to Unity by calling this interface. The Unity will first validate the both OTPs and then change the old email with the new email address.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/email/changeconfirm	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
Request Body	{ "user_id": "johnDoe12", "sms_otp": "225665", "email_otp": "654456", "profile_id": "profile-001" }

	If only one OTP will be received by user either on SMS or email, the request would contain only “sms_otp” or “email_otp”.	
Status Code	**Message**	**Response Body**
200	OK	
400	Bad Request	
401	Unauthorized	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 12 – Confirm Change User Email

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
sms_otp	MANDATORY	*String*	OTP received by the user’s registered mobile number (max. 100 characters).
email_otp	MANDATORY	*String*	OTP received by the user’s registered email (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

3.1.13 Change User Mobile

A business application will call this interface of the ADSS Unity Service to change a user’s mobile number. Like the ‘Change Email’ process, this also completes in two steps. In the first step, the business application sends the user ID and new mobile number to this interface, and ADSS Unity Service will send OTP(s) based on the OTP_TYPE provided in the header. The user can specify either SMS_OTP, EMAIL_OTP, or both. If OTP_TYPE is not specified, OTPs will be sent to both the user’s email and the provided new mobile number by default. After receiving the OTPs, the business application will call another interface, discussed in the next section, to complete the process.

Note: The user’s mobile number can also be changed using the “Update User” API, but this API provides more security and control and clients can choose the preferred API according to the requirements.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/mobile/change	
HTTP Verb	POST

Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
OTP_TYPE header	This header tells the UNITY Service what type of OTP(s) to generate. It can contain one of the following values: • SMS_OTP • EMAIL_OTP • SMS_OTP, EMAIL_OTP **Note:** These values are not case sensitive and clients can send "sms_otp" or "email_otp" as well. Similarly, the name of the header OTP_TYPE is also case insensitive and otp_type will also be accepted by UNITY.

Request Body	{
	"user_id": "johnDoe12",
	"user_mobile": "+448007720442",
	"profile_id": "profile-001"
	}

Status Code	Message	Response Body

200	OK	If two OTPs will be sent to user:
		[
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		},
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}
		]

		If one OTP will be sent on user email:
		{
		"type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		}

		If one OTP will be sent to user's mobile:
		{
		"type": "SMS_OTP",
		"sent_to": "+448007720442"
		}

400	Bad Request	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 13 – Change User Mobile

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	UNITY Profile ID that will be used to process the request (max. 50 characters).
user_mobile	MANDATORY	*String*	New mobile number for the registered user (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
type	MANDATORY	*String*	As OTP can be sent on both mediums i.e email/mobile so it defines the type (max. 100 characters).
sent_to	MANDATORY	*String*	It could be the mobile number or email of the user depends upon the OTP type (max. 500 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.14 Confirm Change User Mobile

Once the OTPs to change user mobile are received by the user. The business application will call this interface providing the both OTPs to Unity Service. The Unity Service will first validate the OTPs and after successful verification, it will change the old mobile number with the new one.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/mobile/changeconfirm	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
Request Body	{ "user_id": "johnDoe12", "sms_otp": "225665", "email_otp": "654456", "profile_id": "profile-001" }

	If only one OTP will be received by user either on SMS or email, the request would contain only “sms_otp” or “email_otp”.	
Status Code	**Message**	**Response Body**
200	OK	
400	Bad Request	
401	Unauthorized	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 14 – Confirm Change User Mobile

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
sms_otp	MANDATORY	*String*	OTP received by the user’s registered mobile number (max. 100 characters).
email_otp	MANDATORY	*String*	OTP received by the user’s registered email (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.15 Get Registered Devices

This API is used to get a list of all the registered mobile devices of a user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/devices/{user_id}?{profile_id}=xyz	
HTTP Verb	GET
Content-Type	
Accept	application/json

Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	[
		{
		"device_id": "2eb1846d-81d8-40d0-86ba-d20bdf7ac5e0",
		"device_name": "IPhone",
		"secure_element": true,
		"biometric": true,
		},
		{
		"device_id": "3fc29573-92e9-40d0-86ba-d20bdf7ac5e0",
		"device_name": "Samsung",
		"secure_element": true,
		"biometric": true,
		
		}
		]

404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 15 – Get Registered Devices

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
profile_id	OPTIONAL	*String*	UNITY Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
device_id	MANDATORY	*String*	Device ID which is created at the time of device registration (max. 255 characters).
device_name	MANDATORY	*String*	Device name which is set at the time of device registration (max. 255 characters).
secure_element	MANDATORY	*Boolean*	“True” if device has secure element/enclave.

biometric	MANDATORY	*Boolean*	"True" if device has biometric feature available on the device. It can be TouchID, FaceID, Fingerprint etc.
user_id	MANDATORY	*String*	User ID identifying the user the device belongs to (max. 50 characters).
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.16 Delete Device

Deletes a user's mobile device in UNITY Service identified by {*user_id*} and {*device_id*}.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/users/devices/{user_id}/{device_id}?{profile_id}=xyz		
HTTP Verb	DELETE	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 16 – Delete Device

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
{user_id}	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
{device_id}	MANDATORY	*String*	Device ID identifying the mobile device (max. 255 characters).
{profile_id}	OPTIONAL	*String*	UNITY Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.17 Generate Key Pair

Creates a key pair for the user in Untiy Service. This key pair will be used to sign the documents. When a new key pair is created the response status '201' is returned.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/keypairs		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body	{ "user_id": "johnDoe12", "app_name": "Application_01", "user_password": "*****", "key_alias": "sample_key_alias", "profile_id": "adss:unity:profile:001" }	
Status Code	**Message**	**Response Body**
201	Created	
200	OK	
400	Bad Request	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 17 –Generate Key Pair

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user for whom the key pair is being generated (max. 50 characters).
app_name	OPTIONAL	*String*	Name of the organization that requested the client's Business Application to generate the key-pair for a user. The Business Application can be dealing with multiple organizations so it can store the actual organization's information with the key-pair to identify keys of a particular organization (max. 50 characters).

user_password	OPTIONAL	*String*	Password will only be required if key wrapping with Dynamic KEK is enabled in Hardware Crypto Profile (max. 500 characters).
key_alias	MANDATORY	*String*	Key Alias to identify the key pair (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.18 Delete Key Pair

Deletes a user's keypair in Unity Service identified by {*user_id*} and {*key_alias*}. The business applications will call this interface to delete a key-pair of a user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/keypairs/{user_id}/{key_alias}?profile_id={profile_id}		
HTTP Verb	DELETE	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 18 – Delete Key Pair

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
{user_id}	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
{key_alias}	MANDATORY	*String*	Key Alias of the key pair that is going to be deleted (max. 50 characters).

{profile_id}	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.19 Get CSR

Returns the base64 encoded CSR (Certificate Signing Request i.e. PKCS#10) of the key pair generated for a user. The business applications will call this interface to get a CSR after generating a key-pair for a user. The client will get this CSR certified and provide the certificate to Unity Service using the "Import Certificate" API discussed next.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/keypairs/csr	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}

Request Body	{
	"user_id": "johnDoe12",
	"user_password": "password12",
	"key_alias": "sample_key_alias",
	"profile_id": "profile-001",
	"certification_request_info ": {
	"subject": [
	{
	"rdn": "CN",
	"rdn_value": ["John Doe"]
	},
	{
	"rdn": "O",
	"rdn_value": ["Example Corp"]
	},
	{
	"rdn": "C",
	"rdn_value": ["US"]
	}
	],
	"attributes": {
	"key_usage": {
	"critical": true,
	"value": ["digitalSignature", "nonRepudiation"]
	},

```
      "ext_key_usage": {
        "critical": true,
        "value": ["clientAuth", "emailProtection"]
      },
      "san": {
        "critical": true,
        "directory_name": [
          {
            "rdn": "CN",
            "rdn_value": ["John Doe"]
          },
          {
            "rdn": "0",
            "rdn_value": ["Example Corp"]
          },
          {
            "rdn": "C",
            "rdn_value": ["US"]
          }
        ],
        "dns_name": ["example.com", "www.example.com"],
        "edi_party_name": [
          {
            "name": "1.2.3.4.5.0",
            "name_encoding": "UTF8String",
            "value": "someValue",
            "value_encoding": "UTF8String"
          }
        ],
        "ip_address": ["192.168.1.100"],
        "other_name": [
          {
            "encoding": "UTF8String",
            "oid": "1.2.3.4.5.0",
            "value": "OtherNameValue"
          }
        ],
        "registered_id": ["1.2.3.4.5.6"],
        "rfc822_name": ["john@example.com"],
        "uniform_resource_identifier":["<https://example.com>"]
      }
    }
}

}
```


Status Code	Message	Response Body
200	OK	{ "csr": " MIICUzCCATsCAQAwDjEMMAoGA1……KJh" }
400	Bad Request	

404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 19 – Get CSR

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
user_password	OPTIONAL	*String*	Used to unwrap the user key if the key was wrapped with a dynamic KEK during generation (max. 500 characters).
key_alias	MANDATORY	*String*	Key Alias of key pair for which CSR to be generated (max. 50 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
certification_request_info	OPTIONAL	*Object*	Additional Certificate Request Info that includes Subject, Extensions and SAN.

Response Parameters

Parameters	Presence	Value	Description
csr	MANDATORY	*String*	Base 64 encoded CSR.
error_code	MANDATORY	*String*	The error code.
error_description	MANDATORY	*String*	Error description message.

3.1.20 Import Certificate

Uploads or import the user's certificate and certificate chain related to a key of the user. This certificate will be stored against its relevant key-pair. A certificate must be imported to Unity Service cause a key cannot be used for signing if not certified, so certificate provides the proof the key has been certified.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/keypairs/cert	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}

Request Body	{
	"user_id": "johnDoe12",
	"key_alias": "sample_key_alias",
	"profile_id": "profile-001",
	"certificate": "HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh=",
	"certificate_chain": [
	"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh=",
	"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh=",
	…
	],
	"p7b": "HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh="
	}

Status Code	Message	Response Body
200	OK	
400	Bad Request	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	

Table 20 – Import Certificate

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
user_id	CONDITIONAL	*String*	User ID identifying the registered user (max. 50 characters).
key_alias	MANDATORY	*String*	Key Alias of key pair for which certificate is being imported.
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request (max. 50 characters).
certificate	MANDATORY	*String*	Base 64 encoded certificate.
certificate_chain	CONDITIONAL	*Array*	Array containing certificates chain in Base 64 encoded string.
p7b	CONDITIONAL	*String*	Certificate chain can also be provided in p7b format as Base 64 encoded string. certificate_chain and p7b can be provided alternatively. If both are present p7b will override certificate_chain.

Response Parameters

Parameters	Presence	Value	Description

error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.1.21 Get User's Certificates

Returns a list of all the certificates (with chains) for the provided registered user.

Exposed for: Business Applications

https://server:8778/adss/service/unity/v1/keypairs/cert/{user_id}?{profile_id}=adss:unity:profile:001		
HTTP Verb	GET	
Content-Type		
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	[{
		"user_id": "Alice",
		"key_alias": "sample_cert_alias_01",
		"app_name": "Application_01"
		"key_status": "ACTIVE",
		"certificate_chain": [
		{"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh="},
		{"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh="}
		…
		]
		},
		{
		"user_id": " Alice ",
		"key_alias": "sample_cert_alias_02",
		"app_name": "Application_01"
		"key_status": " ACTIVE ",
		"certificate_chain": [
		{"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh="},
		{"HyguhugyCATsCAQAwDjEMMAoGA1……jhgjh="}
		…
		]
		}]

404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 21 – Get User's Certificate

Request Parameters

Parameters	Presence	Value	Description
Authorization	MANDATORY	*String*	OAuth access token obtained as a result of successful client or user authentication.
{user_id}	CONDITIONAL	*String*	User ID identifying the registered user in Unity Service (max. 50 characters).
{query_params}	MANDATORY	*String*	Currently supported parameters are: • profile_id • app_name Response will contain the certificates that contain this app_name provided as query parameter e.g. …/service/unity/v1/keypairs/cert/list/user_01?profile_id=xyz&app_name=application01

Response Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID identifying the registered user (max. 50 characters).
app_name	OPTIONAL	*String*	Application name that was provided by business application while generating the key pair (max. 50 characters).
key_alias	MANDATORY	*String*	Key Alias of key pair the certificate belongs to (max. 50 characters).
key_status	MANDATORY	*String*	Status of the key i.e. Active or Inactive (max. 20 characters).
certificate	MANDATORY	*String*	Base 64 encoded string representing the certificate.
certificate_chain	MANDATORY	*Array*	Array containing certificates chain in Base 64 encoded string.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2 CSC APIs

Ascertia has implemented CSC protocol to perform remote authorised signing. The Cloud Signature Consortium (CSC) is a group of industry and academic organizations committed to building new standards for cloud-based digital signatures that will support web and mobile applications and comply with the most demanding electronic signature regulations in the world.

Ascertia Unity Service supports CSC v2 APIs according to the specification version (2.0.0.2). For complete details of the APIs and parameters, please refer to the CSC specification for the same version (2.0.0.2).

3.2.1 Info

This call returns the meta information and the list of endpoints implemented by the service.

https://<server>:8778/adss/service/unity/csc/v2/info		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	No Auth	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	{ "specs": "1.0.3.0", "name": "Ascertia UNITY", "logo": "https://localhost:8777/images/logo.png", "region": "GB", "lang": "en-gb", "description": "UNITY - CSC Service provides remote authorization service implementing protection profiles", "oauth2BaseURI": "http://localhost:8777/adss/service/unity/csc/v1", "authType": ["basic", "oauth2code", " oauth2client "], "methods": ["auth/login", "auth/revoke", "credentials/list", "credentials/info", "credentials/authorize", "signatures/signHash", "oauth2/authorize", "oauth2/token", "oauth2/revoke"], "signAlgorithms": {

		"algos": ["1.2.840.10045.4.3.2", "1.2.840.113549.1.1.1", "1.2.840.113549.1.1.10"]
		},
		"signature_formats": {
		"formats": ["C", "X", "P"],
		"envelope_properties": [["Detached", "Attached", "Parallel"], ["Enveloped", "Enveloping", "Detached"], ["Certification", "Revision"]] },
		"conformance_levels": ["Ades-B-B", "Ades-B-T"]
		}

404	Not Found	HTTP Status 404 – Not Found

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 22 – Info

Request Parameters

Parameters	Presence	Value	Description
lang	OPTIONAL	*String*	Request a preferred language of the response to the remote service, specified according to RFC 5646. If present, the remote service shall provide language-specific responses using the specified language. If the specified language is not supported then it shall provide these responses in the language as specified in the *lang* output parameter.

Response Parameters

Parameters	Presence	Value	Description
specs	MANDATORY	*String*	The version of the specification implemented by the UNITY Service. The format of the string is Major.Minor.x.y, where Major is a number equivalent to the API version (e.g. 1 for API v1) and Minor is a number identifying the version update, while x and y are subversion numbers (max. 2000 characters). The value corresponding to implemented specification is “1.0.3.0” .
name	MANDATORY	*String*	The commercial name of the remote service. The maximum size of the string is 255 characters.
logo	MANDATORY	*String*	The URI of the image file containing the logo of the UNITY Service which shall be published online. The image shall be in either JPEG or PNG format and not larger than 256x256 pixels (max. 2000 characters).

region	MANDATORY	*String*	The ISO 3166-1 Alpha-2 code of the Country where the UNITY Service is established (e.g. ES for Spain) (max. 2000 characters).
lang	MANDATORY	*String*	The language used in the responses, specified according to RFC 5646.
description	MANDATORY	*String*	A free form description of the UNITY Service in the lang language. The maximum size of the string is 255 characters.
oauth2	CONDITIONAL	*String*	The base URI of the OAuth 2.0 authorization server endpoint supported by the remote service for service authorization and/or credential authorization. The parameter shall be present in any of the following cases: • The authType parameter contains “oauth2code” or “oauth2client”; • The remote service supports the value “oauth2code” for the authMode parameter returned by credentials/info. This URI shall be combined with the OAuth 2.0 endpoints (max. 2000 characters).
authType	MANDATORY	*String*	One or more values corresponding to the service authorization mechanisms supported by the remote service to authorize the access to the API (max. 2000 characters). The following mechanisms are supported in UNITY Service: • **Basic:** In case of HTTP Basic Authentication. • **Oauth2code:** In case of OAuth 2.0 with authorization code flow. • **Oauth2client:** In case of OAuth 2.0 with client credentials flow.
methods	MANDATORY	*String*	The list of names of all the API methods described in the specification that are implemented and supported by the UNITY Service (max. 2000 characters).
oauth2Issuer	REQUIRED Conditional	String	The issuer URL of the OAuth 2.0 authorization server as defined in IETF RFC 8414 [23] supported by the remote service for service authorization and/or credential authorization. The parameter SHALL be present if the authType parameter contains “oauth2code” or“oauth2client” or the remote service supports the value “oauth2code” for the auth/mode parameter returned by credentials/info (as specified in credentials/info) and the parameter “oauth2” is not present. The OAuth endpoint URLs are obtained from the
asynchronousOperationMode	OPTIONAL	Boolean	This parameter shall be “true” if the remote signing server supports also asynchronous signature mechanism. The default value is “false”. An omitted parameter or the value

			“false” indicates that the remote signing server manages signature requests only in synchronous operation mode.
validationInfo	OPTIONAL	Boolean	This parameter SHALL be “true” if the remote signing server supports the “validationInfo” response parameter of the method signatures/signDoc in not mandatory cases. An omitted parameter or the value “false” indicates that the remote signing server does not support “validationInfo” in those cases.
signAlgorithms	REQUIRED	JSON Object	Object including one or more signature algorithms supported by the RSSP.
signature_formats	REQUIRED	JSON Object	Object including one or more signature formats supported by the RSSP.
conformance_levels	REQUIRED	Array of String	The list of names of all signature conformance levels supported by the RSSP as defined in the Input parameter table in signatures/signDoc.
algos	REQUIRED	Array of String	The list of signature algorithms supported by the RSSP as defined in the Input parameter table in signatures/signHash. The supported signature algorithms SHOULD follow the recommendations of ETSI TS 119 312 [21] and SHALL be expressed as defined in Expressing algorithms clause.
algoParams	REQUIRED Conditional	Array of String	The list of eventual signature parameters as defined in the Input parameter table in signatures/signHash.

3.2.2 Authentication/Login

It is a username and password based authentication call which after successful authentication returns an access token and optionally refresh token based on input parameter in request.

https://<server>:8778/adss/service/unity/csc/v2/auth/login		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Basic c2FuOnBhc3N3b3Jk… This is the base64 encoded value of Username:UserPassword.	

Request Body	{
	"client_id": “adss…client”,
	"client_secret": “fj49kl…….oOpQS”,
	"profile_id": “ADSS UNITY Profile 001”,
	"rememberMe": true
	
	}	

Status Code	**Message**	**Response Body**

200	OK	{
		"access_token": "eyJhbGciOiJIUzI1NsInN…Pcxcz2hM",
		"refresh_token": "eyJJpc3MinRpYSN1YiI6InN…PCgvAI",
		"expires_in": 3600
		}

400	Bad Request	{
		"error": "58039",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

401	Unauthorised	{
		"error": "59033",
		"error_description": "Failed to process request - user ID or password is invalid"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 23 – Authentication/Login

Request Parameters

Parameters	Presence	Value	Description
client_id	MANDATORY	*String*	The unique *'client_id'* previously assigned to the signature application by remote service (max. 50 characters).
client_secret	MANDATORY	*String*	The *'client_secret'* shall be passed if no authorization header and no client_assertion is used (max. 200 characters).
profile_id	OPTIONAL	*String*	Unity Profile ID that will be used to process the request. This is Ascertia's custom parameter (max. 50 characters).
refresh_token	CONDITIONAL	*String*	The long-lived refresh token returned from a previous call to this method with HTTP Basic Authentication. This may be used as an alternative to the Authorization header to reauthenticate the user according to the method described in RFC 6749. In such case, the encoded userId and password shall not be provided in the HTTP Authorization header. **Note:** This refresh token may not be compatible with refresh tokens obtained by means of OAuth 2.0 authorization.
remember_me	OPTIONAL	*Boolean*	A Boolean value typically corresponding to an option that the user may activate during the authentication phase to 'stay signed in' and maintain a valid authentication across multiple sessions: • **True:** If the remote service supports user reauthentication, a *refresh_token* will be

			returned and the signature application may use it on a subsequent call to this method instead of passing an Authorization header. • **False:** A '*refresh_token*' will not be returned. If the parameter is omitted, it will default to 'false'.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	The short-lived services access token used to authenticate the subsequent API requests within the same session. This token shall be used as the value of the 'Authorization: Bearer' in the HTTP header of the API requests. When receiving an API call with an expired token, the remote services shall return an error and require a new auth/login request.
refresh_token	CONDITIONAL	*String*	The long-lived refresh token used to re-authenticate the user on the subsequent session. The value is returned if the *rememberMe* parameter in the requests is 'true' and the remote service supports user authentication.
expires_in	OPTIONAL	*String*	The lifetime in seconds of the service access token. If omitted, the default expiration time is 3600 seconds i.e. 1 hour (max. 23 characters).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.3 Authentication/Revoke

Revoke a service access token or refresh token that was obtained from the Unity Service. This method exists to enforce the security of the Unity Service. When the Business Application needs to terminate a session, it is recommended to invoke this method to prevent further access by reusing the token.

https://<server>:8778/adss/service/unity/csc/v2/auth/revoke	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {user_access_token}

Request Body	{
	"token": "_TiHRG-bA H3XlFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw",
	"token_type_hint": "refresh_token"
	}

Status Code	Message	Response Body
200	OK	

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid string parameter token_type_hint"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter token"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid string parameter token"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 24 – Authentication/Revoke

Request Parameters

Parameters	Presence	Value	Description
token	MANDATORY	*String*	The token that the signature application wants to get revoked.
token_type_hint	OPTIONAL	*String*	An optional hint about the type of the token submitted for revocation. If the parameter is omitted, the Unity Service will identify the token across all the available tokens (max. 10 characters).
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or

			other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.4 Credentials/List

Returns the list of credentials associated with a user identifier. A user may have one or multiple credentials.

https://<server>:8778/adss/service/unity/csc/v2/credentials/list		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | client_access_token}	

Request Body	<pre>{
	"userID": "Jhon",
	"credentialsInfo": false,
	"credentialID": "JohnDoe",
	"certificates": "chain",
	"certInfo": true,
	"authInfo": true,
	"onlyValid": false,
	"clientData": {
	"certification_profile": "adss:certification:profile:001"
	}
	}</pre>	

Status Code	**Message**	**Response Body**

200	OK	<pre>{
		"credentialIDs": [
		"johnDoe"
		]
		}</pre>

400	Bad Request	<pre>{
		"error": "invalid_request",</pre>

		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed." }

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 25 – Credentials/List

Request Parameters

Parameters	Presence	Value	Description
userID	REQUIRED Conditional	*String*	The identifier associated to the identity of the credential owner. This parameter SHALL NOT be present if the service authorization is user-specific (see NOTE below).In that case the *userID* is already implicit in the service access token passed in the Authorization header.If a user-specific service authorization is present, it SHALL NOT be allowed to use this parameter to obtain the list of credentials associated to a different user. The remote service SHALL return an error in such case.
credentialInfo	OPTIONAL	Boolean	Request to return the main information included in the public key certificate and the public key certificate itself or the certificate chain associated to the credentials. The default value is “false”, so if the parameter is omitted then the information will not be returned.
certificates	OPTIONAL Conditional	String | none |single |chain	Specifies which certificates from the certificate chain SHALL be returned incerts/certificates. “none”: No certificate SHALL be returned. “single”: Only the end entity certificate SHALL be returned. “chain”: The full certificate chain SHALL be returned. The default value is “single”, so if the parameter is omitted then the method will only return the end entity certificate(s).This parameter MAY be specified only if the parameter credentialInfo is “true”. If the parameter credentialInfo is not “true” and this parameter is specified its value SHALL be ignored.
certInfo	OPTIONAL Conditional	Boolean	Request to return various parameters containing information from the end entitycertificate(s). This is useful in case the signature application wants to retrieve somedetails of the certificate(s) without having to decode it first. The default value is“false”, so if the parameter is omitted then the information will not be returned.This parameter MAY be

			specified only if the parameter *credentialInfo* is "true". Ifthe parameter *credentialInfo* is not "true" and this parameter is specified its valueSHALL be ignored.
authInfo	OPTIONAL Conditional	Boolean	Request to return various parameters containing information on the authorization mechanisms supported by the corresponding credential (auth group). The default value is "false", so if the parameter is omitted then the information will not be returned. This parameter MAY be specified only if the parameter credentialInfo is "true". If the parameter credentialInfo is not "true" and this parameter is specified its value SHALL be ignored.
onlyValid	OPTIONAL Conditional	Boolean	Request to return only credentials usable to create a valid signature. The default value is "false", so if the parameter is omitted then the method will return all credentials available to the owner. The remote service MAY NOT support this parameter. When the parameter is supported SHALL be returned in output.
clientData	OPTIONAL	JSON Object	Custom data provided by the signature application. It can be used to reference a certification profile identifier or for other application-specific purposes. **Warning:** This parameter may contain sensitive information, which could be exposed to the remote service. Exercise caution when using it to ensure data security.

Response Parameters

Parameters	Presence	Value	Description
credentialIDs	REQUIRED	Array of String	One or more credentialID(s) associated with the provided or implicit userID.
credentialInfos	OPTIONAL Conditional	Array of Credential Info	The contents of credentialInfo object are described below. If the credentialInfo parameter is not "true", this value SHALL NOT be returned.
onlyValid	REQUIRED Conditional	Boolean	This value SHALL be returned true when the input parameter "onlyValid"was true, and the RSSP supports this feature, i.e. the RSSP only returns credentials which can be used for signing. If the values is false or the output parameter is omitted, then the list may contain credentials which cannot be used for signing.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

The 'credentialInfo Object' is a JSON Object composed by the attributes specified in the following table:

Parameters	Presence	Value	Description
credentialID	MANDATORY	*String*	One or more credentialID(s) associated with the provided or implicit userID. No more than *maxResults* items shall be returned.
description	OPTIONAL	*String*	A free form description of the credential in the *lang* language. The maximum size of the string is 255 characters.
key/status	MANDATORY	*String*	The status of the signing key of the credential are as follows: • **Enabled:** If the signing key is enabled, it can be used for signing. • **Disabled:** If the signing key is disabled, it cannot be used for signing. This may occur when the operator has disabled it or when it has been detected that the associated certificate is expired or revoked. Maximum characters limit is 20.
key/algo	MANDATORY	*String*	The list of OIDs of the supported key algorithms. For example: • 1.2.840.113549.1.1.1=RSA encryption • 1.2.840.10045.4.3.2=ECDSA with SHA256 Maximum characters limit is 10.
key/len	MANDATORY	*Number*	The length of the cryptographic key in bits.
key/curve	CONDITIONAL	*String*	The OID of the ECDSA curve. The value shall only be returned if *keyAlgo* is based on ECDSA (max. 10 characters).
cert/status	OPTIONAL	*String*	The status of validity of the end entity certificate. The value is OPTIONAL, so the UNITY Service will only return a value that is accurate and consistent with the actual validity status of the certificate at the time the response is generated (max. 10 characters).
cert/certificates	CONDITIONAL	*String*	One or more Base64-encoded X.509v3 certificates from the certificate chain. If the certificates parameter is **"chain"**, the entire certificate chain shall be returned with the end entity certificate at the beginning of the array. If the certificates parameter is **"single"**, only the end entity certificate shall be returned. If the certificates parameter is **"none"**, this value shall not be returned.
cert/issuerDN	CONDITIONAL	*String*	The Issuer Distinguished Name from the X.509v3 end entity certificate as UTF-8-encoded character string according to RFC 4514. This value shall be returned when *certInfo* is **"true"**.
cert/serialNumber	CONDITIONAL	*String*	The Serial Number from the X.509v3 end entity certificate represented as hex-encoded string

			format. This value shall be returned when *certInfo* is **"true"**.
cert/subjectDN	CONDITIONAL	*String*	The Subject Distinguished Name from the X.509v3 end entity certificate as UTF-8-encoded character string, according to RFC 4514. This value shall be returned when *certInfo* is **"true"**.
cert/validFrom	CONDITIONAL	*String*	The validity start date from the X.509v3 end entity certificate as character string, encoded as GeneralizedTime (RFC 5280) (e.g. "YYYYMMDDHHMMSSZ"). This value shall be returned when *certInfo* is **"true"**.
cert/validTo	CONDITIONAL	*String*	The validity end date from the X.509v3 end entity certificate as character string, encoded as GeneralizedTime (RFC 5280) (e.g. "YYYYMMDDHHMMSSZ"). This value shall be returned when *certInfo* is **"true"**.
authMode	MANDATORY	*String*	Specifies one of the authorization modes from below: • **Implicit:** The authorization process is managed by the UNITY Service autonomously. Authentication factors are managed by the UNITY by interacting directly with the user, and not by the business application. • **Oauth2code:** The authorization process is managed by the UNITY Service using an OAuth 2.0 mechanism. • **Explicit:** The authorization process is managed by the signature application, which collects authentication factors like PIN or One-Time Passwords (OTP). Maximum characters limit is 20.
SCAL	OPTIONAL	*String*	Specifies if the UNITY Service will generate for this credential a signature activation data (SAD) that contains a link to the hash to-be-signed: • **"1":** The hash to-be-signed is not linked to the signature activation data. • **"2":** The hash to-be-signed is linked to the signature activation data. This value is OPTIONAL and the default value is "1" (max. 500 characters). **NOTE:** The difference between SCAL1 and SCAL2, as described in CEN TS 119 241-1 [i.5], is that for SCAL2, the signature activation data needs to have a link to the data to-be-signed. The value "2" only gives information on the link between the hash and the SAD, it does not give information if a full SCAL2 as described in CEN TS 119 241-1 [i.5] is implemented.

			NOTE: UNITY Service always returns “2” for this parameter.
auth/expression	OPTIONAL Conditional	String	An expression defining the combination of authentication objects required to authorize usage of the private key.If empty, an “AND” of all authentication objects is implied. Supported operators are: “AND” | “OR” | “XOR” | “(” | “)” This value SHALL NOT be returned if auth/mode is not “explicit”.
auth/objects	REQUIRED Conditional	Array of authentication object types	The authentication object types available for this credential. This value SHALL only be returned if auth/mode is “explicit”.
multisign	REQUIRED	Number≥ 1	A number equal or higher to 1 representing the maximum number of signatures that can be created with this credential with a single authorization request (e.g. by calling credentials/ signHash method, as defined in signatures/signHash, once with multiple hash values or callingit multiple times). The value of numSignatures specified in the authorization request SHALL NOT exceed the value of this value.
lang	OPTIONAL	String	The lang as defined in the Output parameter table in info.

3.2.5 Credentials/Info

Retrieve the credential and return the main identity information and the public key certificate or the certificate chain associated to it.

https://<server>:8778/adss/service/unity/csc/v2/credentials/info		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	

Request Body	{
	"credentialID": "JohnDoe",
	"certificates": "chain",
	"certInfo": true,
	"authInfo": true
	}	

Status Code	**Message**	**Response Body**

200	OK	{
		"description": "Go>Sign mobile based implicit credential authorization",
		"key": {
		"status": "enabled",
		"algo":
		[
		"1.2.840.113549.1.1.1"
		],
		"len": 2048,
		"curve": "1.2.840.10045.3.1.7",
		},
		"cert ": {
		"status": "valid",
		"certificates": [
		"MIIFDzCCNI0lRV+Vbe132oyYm1dzz9GI1VqEiPqaKc8miP
		Wb1ssF+MNylYBk\r\n7qnt",
		"MIIFDzCCNI0lRV+Vbe132oyYm1dzz9GI1VqEiPqaKc8miP
		Wb1ssF+MNylYBk\r\n7qnt",
		"MIIEhzCCy2VCpgcCp1xOLhqp+A3TF/8c07EEDjhcmK
		OVEFz\r\nscEsJHTHgj2/buU="
		],
		"issuerDN": "CN=ADSS Samples Test CA,OU=Ascertia Software Distribution,O=Ascertia Limited,C=GB",
		"SerialNumber": "214548948485166938134883584853795496857900189230",
		"subjectDN": "CN=za,OU=Development,O=Ascertia,C=GB",
		"validFrom": "20201215112008+0000",
		"validTo": "20211215112008+0000",
		},
		"auth": {,
		"mode: "explicit"
		"expression": "PIN or OTP",
		"objects": [
		{
		"type": "Password",
		"id": "PIN",
		"label": "PIN",
		"format": "N",
		"description": "Please enter the signature PIN"
		},]
		},
		"SCAL": "2",
		"multisign": 1,
		"lang": "en-GB"
		}

400	Bad Request	{

		"error": "invalid_request", "error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed." }

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter credentialID"
		}

400	Bad Reqeust	{
		"error": "58100",
		"error_description": "Invalid parameter credentialID"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 26 – Credentials/Info

Request Parameters

Parameters	Presence	Value	Description
credentialID	MANDATORY	*String*	The unique identifier associated to the credential (max. 50 characters).
certificates	OPTIONAL	*String*	Specific which certificates from the certificate chain shall be returned in cert/certificates. • **None:** No certificate shall be returned. • **Single:** Only the end entity certificate shall be returned. • **Chain:** The full certificate chain shall be returned. The default value is 'single', so if the parameter is omitted then the method will only return the end entity certificate.
certInfo	OPTIONAL	*Boolean*	Request to return various parameters containing information from the end entity certificate. This is useful in case the business application wants to retrieve some details of the certificate without having to decode it first. The default value is 'false', so if the parameter is omitted then the information will not be returned.
authInfo	OPTIONAL	*Boolean*	Request to return various parameters containing information on the authorization mechanisms supported by this credential. The default value is 'false', so if the parameter is omitted then the information will not be returned.

lang	OPTIONAL	*String*	Request a preferred language of the response to the remote service, specified according to RFC 5646. If present, the remote service shall provide language-specific responses using the specified language. If the specified language is not supported then it shall provide these responses in the language as specified in the lang output parameter.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
description	OPTIONAL	*String*	A free form description of the credential in the *lang* language. The maximum size of the string is 255 characters.
key/status	MANDATORY	*String*	The status of the signing key of the credential are as follows: • **Enabled:** If the signing key is enabled, it can be used for signing. • **Disabled:** If the signing key is disabled, it cannot be used for signing. This may occur when the operator has disabled it or when it has been detected that the associated certificate is expired or revoked. Maximum characters limit is 20.
key/algo	MANDATORY	*String*	The list of OIDs of the supported key algorithms. For example: • 1.2.840.113549.1.1.1=RSA encryption • 1.2.840.10045.4.3.2=ECDSA with SHA256 Maximum characters limit is 10.
key/len	MANDATORY	*Number*	The length of the cryptographic key in bits.
key/curve	CONDITIONAL	*String*	The OID of the ECDSA curve. The value shall only be returned if *keyAlgo* is based on ECDSA (max. 10 characters).
cert/status	OPTIONAL	*String*	The status of validity of the end entity certificate. The value is OPTIONAL, so the UNITY Service will only return a value that is accurate and consistent with the actual validity

			status of the certificate at the time the response is generated (max. 10 characters).
cert/certificates	CONDITIONAL	*String*	One or more Base64-encoded X.509v3 certificates from the certificate chain. If the certificates parameter is **"chain"**, the entire certificate chain shall be returned with the end entity certificate at the beginning of the array. If the certificates parameter is **"single"**, only the end entity certificate shall be returned. If the certificates parameter is **"none"**, this value shall not be returned.
cert/issuerDN	CONDITIONAL	*String*	The Issuer Distinguished Name from the X.509v3 end entity certificate as UTF-8-encoded character string according to RFC 4514. This value shall be returned when *certInfo* is **"true"**.
cert/serialNumber	CONDITIONAL	*String*	The Serial Number from the X.509v3 end entity certificate represented as hex-encoded string format. This value shall be returned when *certInfo* is **"true"**.
cert/subjectDN	CONDITIONAL	*String*	The Subject Distinguished Name from the X.509v3 end entity certificate as UTF-8-encoded character string, according to RFC 4514. This value shall be returned when *certInfo* is **"true"**.
cert/validFrom	CONDITIONAL	*String*	The validity start date from the X.509v3 end entity certificate as character string, encoded as GeneralizedTime (RFC 5280) (e.g. "YYYYMMDDHHMMSSZ"). This value shall be returned when *certInfo* is **"true"**.
cert/validTo	CONDITIONAL	*String*	The validity end date from the X.509v3 end enity certificate as character string, encoded as GeneralizedTime (RFC 5280) (e.g. "YYYYMMDDHHMMSSZ"). This value shall be returned when *certInfo* is **"true"**.
authMode	MANDATORY	*String*	Specifies one of the authorization modes from below: • **Implicit:** The authorization process is managed by the UNITY Service autonomously. Authentication factors are managed by the UNITY by interacting directly with the user, and not by the business application. • **Oauth2code:** The authorization process is managed by the UNITY Service using an OAuth 2.0 mechanism. • **Explicit:** The authorization process is managed by the signature application, which collects authentication factors like PIN or One-Time Passwords (OTP).

			Maximum characters limit is 20.
SCAL	OPTIONAL	*String*	Specifies if the UNITY Service will generate for this credential a signature activation data (SAD) that contains a link to the hash to-be-signed: • **"1":** The hash to-be-signed is not linked to the signature activation data. • **"2":** The hash to-be-signed is linked to the signature activation data. This value is OPTIONAL and the default value is "1" (max. 500 characters). **NOTE:** The difference between SCAL1 and SCAL2, as described in CEN TS 119 241-1 [i.5], is that for SCAL2, the signature activation data needs to have a link to the data to-be-signed. The value "2" only gives information on the link between the hash and the SAD, it does not give information if a full SCAL2 as described in CEN TS 119 241-1 [i.5] is implemented. **NOTE:** UNITY Service always returns "2" for this parameter.
auth/expression	OPTIONAL Conditional	String	An expression defining the combination of authentication objects required to authorize usage of the private key.If empty, an "AND" of all authentication objects is implied. Supported operators are: "AND" | "OR" | "XOR" | "(" | ")" This value SHALL NOT be returned if auth/mode is not "explicit".
auth/objects	REQUIRED Conditional	Array	The authentication object types available for this credential. This value SHALL only be returned if auth/mode is "explicit".
multisign	MANDATORY	*Number*	A number equal or higher to 1 representing the maximum number of signatures that can be created with this credential with a single authorization request.
lang	OPTIONAL	*String*	The language used in the responses, specified according to RFC 5646.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.6 Credentials/Authorize

Authorize the access to the credential for remote signing, according to the authorization mechanisms associated to it. This method returns the Signature Activation Data (SAD) required to authorize the signatures/signHash method.

https://<server>:8778/adss/service/unity/csc/v2/credentials/authorize	
HTTP Verb	POST
Content-Type	application/json

Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
Request Body	{ "credentialID": "JohnDoe", "numSignatures": 2, "documents": [{ "document_id": 123, "document_name": “Document Name 123”, },{ "document_id": 456, "document_name": “Document Name 456”, }], "hashes": ["sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI=", "c1RPZ3dPbSs0NzRnRmowcTB4MWlTTnNwS3FiY3NlNEllaXFsRGcvSFd1ST0="], "hashAlgorithmOID": 2.16.840.1.101.3.4.2.1, “authData” : [{ “id” : “PIN” “value” : “12345678”, }] }, **Note:** The ‘documents’, ‘document_id’ and ‘document_name’ are Ascertia’s custom parameters and are optional in this case.

Status Code	Message	Response Body
200	OK	{ "SAD": "_TiHRG-bA4/CKN69L8gdSYp5_pw" }
400	Bad Request	{ "error": "invalid_request", "error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed." }
400	Bad Request	{ "error": "invalid_request", "error_description": "Missing (or invalid type) string parameter credentialID" }

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter credentialID"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) integer parameter numSignatures"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter numSignatures"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid PIN parameter - Failed to authenticate PIN"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid OTP parameter - Failed to authenticate OTP"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 27 – Credentials/Authorize

Request Parameters

Parameters	Presence	Value	Description
credentialID	MANDATORY	*String*	The unique identifier associated to the credential (max. 50 characters).
numSignatures	MANDATORY	*Number*	The number of signatures to authorize.
documents/document_id	OPTIONAL	*String*	It is Ascertia's custom parameter that will represent the unique ID for the document.
documents/document_name	OPTIONAL	*String*	It is Ascertia's custom parameter that will represent the name of the document.
hash	CONDITIONAL	*String*	One or more Base64-encoded hash values to be signed. It allows the server to bind the SAD to the hash(es), thus preventing an authorization to be used to sign a different content. If the SCAL parameter returned by credentials/info method, for the current credentialID is "2", the hash parameter shall be used and the number of hash values should correspond to the value in

			numSignatures. If the SCAL parameter is “1”, the hash parameter is OPTIONAL.
hashAlgorithmOID	REQUIRED Conditional	String	String containing the OID of the hash algorithm used to generate the hashes.
authData	REQUIRED Conditional	Array	The authentication objects as described by the authentication object types in credentials/info. It SHALL be used only when auth/mode from credentials/info is “explicit”.
description	OPTIONAL	*String*	A free form description of the authorization transaction in the *lang* language. The maximum size of the string is 5000 characters. It can be useful when *authMode* from credentials/info method is “implicit” to provide some hints about the occurring transaction.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.

Response Parameters

Parameters	Presence	Value	Description
SAD	MANDATORY	*String*	The Signature Activation Data (SAD) in base64 encoded format or as JSON Web Signatures depending upon the SAD_FORMAT set in Global Setting > Advanced Settings > SAM to be used as input to the **signatures/signHash** method.
expiresIn	OPTIONAL	*Number*	The lifetime in seconds of the SAD. If omitted, the default expiration time is 3600 (1 hour).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.7 Credentials/Authorizecheck

After a credentials/authorize with HTTP result code 202, the client may use the handle returned to poll the authorization state.

https://<server>:8778/adss/service/unity/csc/v2/authorizeCheck	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json

Authorization	Bearer {client_access_token | user_access_token}	

Request Body	{
	"handle":"sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI="
	}	

Status Code	**Message**	**Response Body**

200	OK	{
		"SAD":"KeTob5gl26S2tmXjqN…MRGtoew=
		="
		}

202	OK	{
		"handle":"sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI="
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid handle parameter"
		}

400	Credential Locked	{
		"error": "invalid_request",
		"error_description": "Credentials Locked"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 28 – Credentials/Authorizecheck

Request Parameters

Parameters	Presence	Value	Description
handle	REQUIRED	String	The handle value returned from credentials/authorize.

Response Parameters

Parameters	Presence	Value	Description
SAD	REQUIRED	*String*	The new Signature Activation Data required to sign multiple times with a single authorization. The Signature Activation Data (SAD) in base64 encoded format or as JSON Web Signatures

			depending upon the SAD_FORMAT set in Global Setting > Advanced Settings > SAM
expiresIn	OPTIONAL	*Number*	The lifetime in seconds of the SAD. If omitted, the default expiration time is 3600 (1 hour).

With HTTP status code 202 the method indicates that some authorization is still underway. The result contains a handle that can be used to poll the state of the authorization via repeated calls to credentials/authorizeCheck.

Parameters	Presence	Value	Description
handle	REQUIRED	String	An opaque handle that can be used to request the state of the authorization.

3.2.8 Credentials/GetChallenge

Get a challenge for a referenced authentication object.

https://<server>:8778/adss/service/unity/csc/v2/getChallenge		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	
Request Body	{ "credentialID":"GX123", "authObjectID":"fallback question" }	
Status Code	**Message**	**Response Body**
200	OK	{ "challenge":"whats your pet name" }
400	Bad Request	{ "error": "invalid_request", "error_description": "Malformed authorization header." }
400	Bad Request	{ "error": "invalid_request", "error_description": "Invalid parameter credentialID" }
400	Bad Request	{ "error": "invalid_request", "error_description": "Invalid parameter authObjectID" }

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 29 – Credentials/GetChallenge

Request Parameters

Parameters	Presence	Value	Description
credentialID	REQUIRED	String	The identifier associated to the credential.
authObjectID	REQUIRED	String	The identifier of the authentication object we need a challenge for.

Response Parameters

Parameters	Presence	Value	Description
challenge	REQUIRED	String	The authentication object challenge.

3.2.9 Credentials/extendTransaction

Extends the validity of a multi-signature transaction authorization by obtaining a new Signature Activation Data (SAD). This method SHALL be used in case of multi-signature transaction when the API method signatures/signHash is invoked multiple times with a single credential authorization event.

https://<server>:8778/adss/service/unity/csc/v1/credentials/extendTransaction		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {access_token}	
Request Body	{ "credentialID": "JohnDoe", "SAD": "_TiHRG-bAH3XlFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw", "hash": ["sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI="],	
Status Code	**Message**	**Response Body**

200	OK	{
		"SAD":"KeTob5gl26S2tmXjqN…MRGtoew=="
		}

400	Bad Request	{ "error": "invalid_request",

		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed." }

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter SAD"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter SAD"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter credentialID
		"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter credentialID"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) array parameter hash"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid Base64 hash string parameter"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid digest value length"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 30 – Credentials/extendTransaction

Request Parameters

Parameters	Presence	Value	Description
credentialID	REQUIRED	*String*	The unique identifier associated to the credential (max. 50 characters).
hash	REQUIRED Conditional	*Array of String*	One or more Base64-encoded hash values to be signed. It allows the server to bind the new SAD to the hash, thus preventing an authorization to be used to sign a different content. It SHALL be used if the SCAL parameter returned by credentials/info for the

			current credentialID is “2”, otherwise it is OPTIONAL.
SAD	REQUIRED	*String*	The current unexpired Signature Activation Data. This token is returned by the credentials/authorize or by the previous call to credentials/extendTransaction. The Signature Activation Data (SAD) in base64 encoded format or as JSON Web Signatures depending upon the SAD_FORMAT set in Global Setting > Advanced Settings > SAM.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.

Response Parameters

Parameters	Presence	Value	Description
SAD	REQUIRED	*String*	The new Signature Activation Data required to sign multiple times with a single authorization. The Signature Activation Data (SAD) in base64 encoded format or as JSON Web Signatures depending upon the SAD_FORMAT set in Global Setting > Advanced Settings > SAM
expiresIn	OPTIONAL	*Number*	The lifetime in seconds of the SAD. If omitted, the default expiration time is 3600 (1 hour).

3.2.10 Signatures/SignHash

Calculate the remote digital signature of one or multiple hash values provided as an input. This method requires providing credential authorization in the form of Signature Activation Data (SAD).

https://<server>:8778/adss/service/unity/csc/v2/signatures/signHash	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
Request Body	{ "credentialID": "JohnDoe", "SAD": "_TiHRG-bAH3XlFQZ3ndFhkXf9P24/CKN69L8gdSYp5_pw", "documents": [{ "document_id": 123, "document_name": “Document Name 123”, },{ "document_id": 456, "document_name": “Document Name 456”,

	}], "hashes": ["sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI="], "hashAlgorithmOID": 2.16.840.1.101.3.4.2.1, "signAlgo": "1.2.840.113549.1.1.1" “signAlgoParams": "2.16.840.1.101.3.4.2.1", "operationMode": "A", } **Note:** The ‘documents’, ‘document_id’ and ‘document_name’ are Ascertia’s custom parameters and are optional in this case.

Status Code	Message	Response Body

200	OK	{
		"signatures":
		[
		"KeTob5gl26S2tmXjqN…MRGtoew=="
		]
		}

200	OK	{
		"responseID":"1234567890"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter SAD"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Expired SAD"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter credentialID "
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter credentialID"
		}

400	Bad Request	{ "error": "invalid_request", "error_description": "Missing (or invalid type) array parameter hash"

		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Empty hash array"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid Base64 hash string parameter"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter signAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter hashAlgorithmsOID"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter hashAlgorithmsOID "
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter signAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid digest value length"
		}

400	Bad Request	{
		"error": "invalid_otp",
		"error_description": "The OTP is invalid"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Signing certificate 'O=[organization],CN=[common_name]' is expired."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter clientData "
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 31 – Signatures/SignHash

Request Parameters

Parameters	Presence	Value	Description
credentialID	MANDATORY	*String*	The unique identifier associated to the credential (max. 50 characters).
SAD	REQUIRED Conditional	String	The Signature Activation Data returned by the Credential Authorization methods. Not needed if the signing application has passed an access token in the "Authorization" HTTP header with scope "credential", which is also good for the credential identified by credentialID. Note: For backward compatibility, signing applications MAY pass access tokens with scope "credential" in this parameter.
documents/document_id	OPTIONAL	*String*	It is Ascertia's custom parameter that will represent the unique ID for the document.
documents/document_name	OPTIONAL	*String*	It is Ascertia's custom parameter that will represent the name of the document.
hashes	MANDATORY	*String*	One or more hash values to be signed. This parameter shall contain the Base64-encoded raw message digest(s).
hashAlgorithmsOID	CONDITIONAL	*String*	The OID of the algorithm used to calculate the hash value(s). This parameter shall be omitted or ignored if the hash algorithm is implicitly specified by the *signAlgo* algorithm. Only hashing algorithms as strong or stronger than SHA256 shall be used. The hash algorithm should follow the recommendations of ETSI TS 119 312 (max. 10 characters).
signAlgo	MANDATORY	*String*	The OID of the algorithm to use for signing. It shall be one of the values allowed by the credential as returned in *keyAlgo* by the credentials/info method (max. 10 characters).
signAlgoParams	CONDITIONAL	*String*	The Base64-encoded DER-encoded ASN.1 signature parameters, if required by the signature algorithm. Some algorithms like RSASSA-PSS, as defined in RFC 8917, may require additional parameters (max. 100 characters).
operationMode	OPTIONAL	String	The type of operation mode requested to the remote signing server. It SHALL take one of the following values: "A": an asynchronous operation mode is requested. "S": a synchronous operation mode is requested.

			The default value is “S”, so if the parameter is omitted then the remote signing server will manage the request in synchronous operation mode.
validity_period	OPTIONAL Conditional	Integer	Maximum period of time, expressed in milliseconds, until which the server SHALL keep the request outcome(s) available for the client application retrieval. This parameter MAY be specified only if the parameter operation Mode is “A”. If the parameter operation Mode is not “A” and this parameter is specified its value SHALL be ignored. The RSSP SHOULD define in its service policy the default and maximum values of this parameter. If the RSSP does not define in its service policy any default and maximum values of this parameter it means that any value MAY be passed in this parameter
response_uri	OPTIONAL Conditional	String	Value of one location where the server will notify the signature creation operation completion, as an URI value. This parameter MAY be specified only if the parameter operation Mode is “A”. If the parameter operation Mode is not “A” and this parameter is specified its value SHALL be ignored. If the parameter operationMode is “A” and this parameter is omitted then the remote signing server will not make any notification.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.

Response Parameters

Parameters	Presence	Value	Description
signatures	MANDATORY	*String*	One or more Base64-encoded signed hash(es). In case of multiple signatures, the signed hash(es) shall be returned in the same order as the corresponding hashes provided as an input parameter.
responseID	REQUIRED Conditional	String	Arbitrary string value generated by the server uniquely identifying the response originated from the server itself. This value SHALL be returned when operationMode is“A”.

error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message

3.2.11 Signatures/SignDoc

Create one or more AdES signatures. Either the documents to be signed or the SDRs (in this specification it is intended to be the hash values of the documents to be signed) SHALL be provided to the method. An AdES signature will be created for each of these input components. Other components are used to select the type of signature that will be created for each document or document representation..

https://<server>:8778/adss/service/unity/csc/v2/signatures/signDoc	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {client_access_token | user_access_token}
Request Body	{ "credentialID": "testKey", "SAD": "PEF1...", "documents": [{ "document": "JVBERi0xLjUNCiW1tbW1DQoxIDAgb2Jq...", "signature_format": "P", "lock_pdf_after_signing": false, "conformance_level": "AdES-B-LT", "signAlgo": "1.2.840.113549.1.1.1", "signed_props": [{ "attribute_name": "Location", "attribute_value": "UK" }, { "attribute_name": "ContactInfo", "attribute_value": "jhon@gmail.com"

<table>
<tr><td></td><td>
<pre>
        },
        {
          "attribute_name": "Name",
          "attribute_value": "Jhon "
        },
        {
          "attribute_name": "Reason",
          "attribute_value": "Casual"
        }
      ]
    }
  ],
  "returnValidationInfo": true,
  "empty_sig_field": "Signature2",
  "m_bDefaultProfile": false,
  "m_bDefaultClient": false
}
</pre>
Note: The ‘documents’, ‘document_id’ and ‘document_name’ are Ascertia’s custom parameters and are optional in this case.
</td></tr>
<tr><td></td><td></td></tr>
</table>

Status Code	Message	Response Body

200	OK	{
		"signatures":
		[
		"KeTob5gl26S2tmXjqN…MRGtoew=="
		]
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter SAD"
		}

400	Bad Request	{

		"error": "invalid_request", "error_description": "Invalid parameter SAD" }

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter credentialID "
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter credentialID"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) array parameter hash"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Empty hash array"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid Base64 hash string parameter"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter signAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Missing (or invalid type) string parameter hashAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter hashAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter signAlgo"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid digest value length"
		}

400	Bad Request	{
		"error": "invalid_otp",
		"error_description": "The OTP is invalid"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Signing certificate 'O=[organization],CN=[common_name]' is expired."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter clientData "
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 32 – Signatures/SignDoc

Request Parameters

Parameters	Presence	Value	Description
credentialID	MANDATORY	*String*	The unique identifier associated to the credential (max. 50 characters).
signatureQualifier	REQUIRED Conditional	String	Identifier of the signature type to be created, e.g. “eu_eidas_qes” to denote a Qualified Electronic Signature according to eIDAS. This specification defines set of such identifiers (see table below), service providers can also define and use their own identifiers. At least one of the two values credentialID and signatureQualifier SHALL be present. Both values MAY be present.
SAD	MANDATORY	*String*	The Signature Activation Data returned by the Credential Authorization methods. The Signature Activation Data (SAD) in base64 encoded format or as JSON Web Signatures depending upon the SAD_FORMAT set in Global Setting > Advanced Settings > SAM.
documents/document_id	OPTIONAL	*String*	It is Ascertia’s custom parameter that will represent the unique ID for the document.
documents/document_name	OPTIONAL	*String*	It is Ascertia’s custom parameter that will represent the name of the document.
documents/pdf_protected_password	CONDITIONAL	*String*	If document is password protected then this parameter will be used to send the PDF password.
hash	MANDATORY	*String*	One or more hash values to be signed. This parameter shall contain the Base64-encoded raw message digest(s).
hashAlgo	CONDITIONAL	*String*	The OID of the algorithm used to calculate the hash value(s). This

			parameter shall be omitted or ignored if the hash algorithm is implicitly specified by the *signAlgo* algorithm. Only hashing algorithms as strong or stronger than SHA256 shall be used. The hash algorithm should follow the recommendations of ETSI TS 119 312 (max. 10 characters).
signAlgo	MANDATORY	*String*	The OID of the algorithm to use for signing. It shall be one of the values allowed by the credential as returned in *keyAlgo* by the credentials/info method (max. 10 characters).
signAlgoParams	CONDITIONAL	*String*	The Base64-encoded DER-encoded ASN.1 signature parameters, if required by the signature algorithm. Some algorithms like RSASSA-PSS, as defined in RFC 8917, may require additional parameters (max. 100 characters).
validity_period	OPTIONAL Conditional	Integer	Maximum period of time, expressed in milliseconds, until which the server SHALL keep the request outcome(s) available for the client application retrieval. This parameter MAY be specified only if the parameter operation Mode is “A”. If the parameter operation Mode is not “A” and this parameter is specified its value SHALL be ignored. The RSSP SHOULD define in its service policy the default and maximum values of this parameter. If the RSSP does not define in its service policy any default and maximum values of this parameter it means that any value MAY be passed in this parameter
response_uri	OPTIONAL Conditional	String	Value of one location where the server will notify the signature creation operation completion, as an URI value. This parameter MAY be specified only if the parameter operation Mode is “A”. If the parameter operation Mode is not “A” and this parameter is specified its value SHALL be ignored. If the parameter operationMode is “A” and this parameter is omitted then the remote signing server will not make any notification.
validity_period	OPTIONAL Conditional	Integer	Maximum period of time, expressed in milliseconds, until which the server SHALL keep the request outcome(s) available for the client application retrieval. This parameter MAY be specified only if the parameter operation Mode is “A”. If the parameter operation

			Mode is not “A” and this parameter is specified its value SHALL be ignored. The RSSP SHOULD define in its service policy the default and maximum values of this parameter. If the RSSP does not define in its service policy any default and maximum values of this parameter it means that any value MAY be passed in this parameter
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.
returnValidationInfo	OPTIONAL	Boolean	This parameter SHALL be set to “true” to request the service to return the“validationInfo” as defined below. The default value is “false”, i.e. no“validationInfo” info is provided. This parameter SHALL be supported in conjunction with “signature_format”“P”, “conformance_level” “AdES-B-LT” and use of “documentDigests”. For all other cases, the info methods states if this feature is supported or not.
hashes	REQUIRED Conditional	Array of String	One or more hash values representing one or more SDRs. This parameter SHALL contain the Base64-encoded hash(es) of the documents to be signed. In case a hashes were provided for the credential authorization, then the RSSP SHALL verify that each of the hashes in this parameter corresponds to one of the hashes provided in the credential authorization.
hashAlgorithmOID	REQUIRED Conditional	String	Hashing algorithm OID used to calculate document(s) hash(es). This parameter MAY be omitted or ignored if the hash algorithm is implicitly specified by the signAlgo algorithm. Only hashing algorithms as strong or stronger than SHA256 SHALL be used. The hash algorithm SHOULD follow the recommendations of ETSI TS 119 312 [21].
signature_format	REQUIRED	String	The required signature format: “C” SHALL be used to request the creation of a CAdES signature; “X” SHALL be used to request the creation of a XAdES signature. “P” SHALL be used to request the creation of a PAdES signature.

conformance_level	OPTIONAL	String	The required signature conformance level: “Ades-B-B” SHALL be used to request the creation of a baseline 191x2 level B signature; “Ades-B-T” SHALL be used to request the creation of a baseline 191x2 level T signature; “Ades-B-LT” SHALL be used to request the creation of a baseline 191x2 level LT signature; “Ades-B-LTA” SHALL be used to request the creation of a baseline 191x2 level LTA signature; “Ades-B” SHALL be used to request the creation of a baseline etsi level B signature; “Ades-T” SHALL be used to request the creation of a baseline etsi level T signature; “Ades-LT” SHALL be used to request the creation of baseline etsi level LT signature; “Ades-LTA” SHALL be used to request the creation of baseline etsi level LTA signature. The parameter is optional. The default level is AdES-B-B in case it is omitted. If a timestamp is needed its request and inclusion is managed by the signing server according to signing server configuration and policies.
signAlgo	REQUIRED	String	The signAlgo as defined in the Input parameter table in signatures/signHash. If the parameter hashAlgorithmOID defined in the documentDigests Object is passed and is in contradiction with the value of this parameter signAlgo the method SHALL return an error condition.
signAlgoParams	REQUIRED Conditional	String	The signAlgoParams as defined in the Input parameter table in signatures/signHash.
signed_props	OPTIONAL	Array of attribute	List of signed attributes. The attributes that may be included dependon the signature format and the signature creation policy. The contents of attribute object are described below.
signed_envelope_property	OPTIONAL Conditional	String	The required property concerning the signed envelope whose possible values depend on the value of the signature_formatparameter.According to the type of selected signature_format a clientapplication may specify the following signature properties. CAdESDetached Attached Parallel

			PAdESCertification Revision XAdESEnveloped Enveloping Detached JAdESDetached Attached Parallel The default values are the following ones. CAdESAttached PAdESCertification XAdESEnveloped JAdESAttached
lock_pdf_after_signing	OPTIONAL	Boolean	This parameter SHALL be set to “true” to request the service to lock the pdf document after signing.

Response Parameters

Parameters	Presence	Value	Description
DocumentWithSignature	REQUIREDConditional	ArrayofString	One or more Base64-encoded signatures enveloped within the documents. This element SHALL carry a value only if the client application requested the creation of signature(s) enveloped within the signed document(s) and when operationMode is not “A”.
SignatureObject	REQUIREDConditional	ArrayofString	One or more Base64-encoded signatures detached from the documents. This element SHALL carry a value only if the client application requested the creation of detached signature(s) and when operationMode is not “A”.
responseID	REQUIREDConditional	String	The responseID as defined in the Output attribute table in signatures/signHash.
validationInfo	REQUIREDConditional	JSONObject	The validationInfo is a JSON Object containing validation data that SHALL be included in the signing response if requested using the input parameter “returnValidationInfo”.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message

3.2.12 Signatures/Timestamp

Generate a time-stamp token for the input hash value. The time-stamp token can be generated directly by the RSSP or by a Time Stamping Authority connected to it.

https://<server>:8778/adss/service/unity/csc/v2/timestamp		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {client_access_token | user_access_token}	

Request Body	{
	"hash":" MTIzNDU2Nzg5MHF3ZXJ0enVpb3Bhc2RmZ2hqa2zDtnl4",
	"hashAlgo":"2.16.840.1.101.3.4.2.1"
	}	

Status Code	**Message**	**Response Body**

200	OK	{
		"timestamp":"Abcfheimidmvd…divnidiwenov
		n"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Malformed authorization header."
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid hash length"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "Invalid parameter hashAlgo"
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 33 – Signatures/Timestamp

Request Parameters

Parameters	Presence	Value	Description
hash	REQUIRED	String	The Base64-encoded hash value to be time stamped. The remote service SHALL use this value to encode the value of

			MessageImprint.hashedMessage as defined in RFC 3161 [2].
hashAlgo	REQUIRED	String	The OID of the algorithm used to calculate the hash value. The remote service SHALL use this value to encode the value of MessageImprint.hashAlgorithm as defined in RFC 3161[2].
nonce	OPTIONAL	String	A large random number with a high probability that it is generated by the signatureapplication only once. The value SHALL be represented as hex-encoded string.
clientData	OPTIONAL	String	The clientData as defined in the Input parameter table in oauth2/authorize.

Response Parameters

Parameters	Presence	Value	Description
timestamp	REQUIRED	String	The Base64-encoded time-stamp token as defined in RFC 3161 [2] as updated by RFC 5816[10]. If the nonce parameter is included in the request then it SHALL also be included inthe time-stamp token, otherwise the response SHALL be rejected.

3.2.13 OAuth2/Authorize

It does not specify a regular CSC API method, but rather the URI path component of the address of the web page allowing the user to sign-in to the remote service to authorize the signature application or to authorize a credential. The complete URL to invoke the OAuth 2.0 authorization server is obtained by adding oauth2/authorize to the base URI of the authorization server as returned in the oauth2 parameter by the “info” method and it does not necessarily include the base URI of the remote service API.

https://<server>:8778/adss/service/unity/csc/v2/oauth2/authorize	
HTTP Verb	GET
Content-Type	
Accept	
Parameters	**//Service Authorization** response_type=code& client_id=samples_test_client& redirect_uri =http://localhost:8777& scope=service& lang=en-UK& state=123456&

	profile_id=adss:unity:profile:001& prompt=login **// Credentials Authorization** response_type=code& client_id=samples_test_client& redirect_uri =http://localhost:8777& scope=credential& credentialID=sample-key& numSignatures=2& hash= MTIzNDU2Nzg5MHF3ZXJ0enVpb3Bhc2RmZ2hqa2zDtnl4& state=12345

Status Code	Message	Response Body
302	Found	Location: <OAuth2_redirect_uri> ? code=12234&state=121212

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 34 – OAuth2/Authorize

Request Parameters

Parameters	Presence	Value	Description
response_type	MANDATORY	*String*	The value shall be “code”.
client_id	CONDITIONAL	*String*	The unique ‘‘client ID’’ previously assigned to the signature application by the UNITY Service. It shall be passed if no account_token is used (max. 50 characters).
redirect_uri	OPTIONAL	*String*	The URL where the user will be redirected after the authorization process has completed. Only

			a valid URI pre-registered with the UNITY Service shall be passed. If omitted, the service will use the default redirect URI pre-registered by the signature application. **Note:** If the Allowed Redirect URI(s) list is configured, the default redirect URI specified in the Redirect URI field must also be added to this list for validation. If the default URI is not included in the allowed list, the authorization request will be rejected, and an error will be returned.
scope	OPTIONAL	*String*	The scopes of the access request are mentioned below: • **Service:** It shall be used to obtain an authorization code suitable for service authorization. • **Credential:** It shall be used to obtain an authorization code suitable for credentials authorization. The parameter is OPTIONAL. The defaults scope is “service” in case it is omitted.
authorization_details	OPTIONAL	*String*	The details of the access request as described in IETF Draft-ietf-oauth-rar [27]. This specification defines the following authorization details type: • “credential”: it SHALL be used to obtain an authorization code suitable for credentials authorization. The parameter is OPTIONAL. If this parameter is used, all values relevant for credential authorization SHALL be passed in this object. The scope “credential” as well as any request parameter relevant for credential authorization SHALL NOT be used in this case.
prompt	OPTIONAL	*String*	When scope=service, the OIDC prompt parameter can be used to specify whether the Identity Provider (IDP) should prompt the end-user for re-authentication or consent. The Unity Service forwards the prompt value to the IDP if it is present in the request. If not present, the default value prompt=login is forwarded to the IDP. When scope=credential, the prompt value is derived from the Unity Profile, and any value provided in the request is ignored. Valid prompt values include: • none • consent • login

			• select_account
code_challenge	MANDATORY	*String*	Cryptographic nonce binding the transaction to a certain user agent, used to detect code replay and CSRF attacks.
code_challenge_method	OPTIONAL	*String*	Code verifier transformation method as defined in IETF RFC7636 [25], defaults to plain. The recommended value is S256.
lang	OPTIONAL	*String*	Request a preferred language according to RFC 5646. If a language is explicitly specified, the authorization server should display the authorization web page in that language, provided it is supported. If no language is specified but an Accept-Language header is included in the request, the authorization server should use the language indicated in the header, if supported. **Note:** The lang parameter in the OAuth 2.0 /authorize request allows business applications to specify the desired language for the authorization page, ensuring it appears in a supported language. If no language preference is provided, the authorization server will default to its pre-configured language setting.
state	OPTIONAL	*String*	Up to 255 bytes of arbitrary data from the signature application that will be passed back to the redirect URI. The use is recommended for preventing cross-site request forgery.
request_uri	CONDITIONAL	*String*	URI pointing to a pushed authorization request previously uploaded by the client. This parameter SHALL only be used in conjunction with the client_id. All other parameters SHALL NOT be combined with this parameter.
profile_id	CONDITIONAL	*String*	It is Ascertia's custom parameter that will represents the UNITY profile ID being used (max. 200 characters). It shall be passed if an account_token is used.
credentialID	CONDITIONAL	*String*	The identifier associated to the credential to authorize. It shall be used only if the scope of the OAuth 2.0 authorization request is "credential". This parameter value may contain characters that are reserved, unsafe or forbidden in URLs and therefore shall be url-encoded by the signature application (max. 50 characters).
numSignatures	CONDITIONAL	*Number*	The number of signatures to authorize. Multi-signature transactions can be obtained by using

			a combination of array of hash values and by calling multiple times the **signatures/signHash** method. It shall be used only if the scope of the OAuth 2.0 authorization request is “credential”.
hashes	CONDITIONAL	*String*	One or more base64url-encoded hash values to be signed. It allows the server to bind the SAD to the hash, thus preventing an authorization to be used to sign a different content. It shall be used if the SCAL parameter returned by credentials/info method, for the current credentialID is “2”, otherwise it is OPTIONAL. Multiple hash values can be passed as comma separated values, e.g. oauth2/authorize?hash=dnN3ZX…ZmRm,ZjIxM3…Z2Zk,… The order of multiple values does not have to match the order of hashes passed to **signatures/signHash** method.
hashAlgorithmOID	CONDITIONAL	*String*	String containing the OID of the hash algorithm used to generate the hashes.
description	OPTIONAL	*String*	A free form description of the authorization transaction in the lang language. The maximum size of the string is 5000 characters. It can be useful to provide some hints about the occurring transaction.
account_token	CONDITIONAL	*String*	An account_token may be required by a RSSP if their authorization server has a restricted access. The value is a JSON Web Token (JWT) according to RFC 7519. It shall be passed if no client_id is used, particularly in cases when there are huge number of clients and each of them cannot be registered in the ADSS server, so instead of passing the client_id, an account_token can be passed. **Note:** The account_token can be generated via OAuth2/Token API by passing the client_assertion and client_assertion_type with grant_type as client_credentials.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.
response_mode	OPTIONAL	*String*	This parameter informs the OP of the mechanism to be used when returning authorization response parameters (like codes or tokens) from the Authorization Endpoint. E.g: query, fragment
display	OPTIONAL	*String*	An ASCII string value that provides a hint to the OP about how it should display the user

			interface pages (authentication and consent screens) to the End-User. E.g: page,wap
max_age	OPTIONAL	*Number*	Description: Specifies the allowable elapsed time (in seconds) since the last time the End-User was actively authenticated by the OpenID Provider. E.g: 0,3600
Id_token_hint	OPTIONAL	*String*	This parameter passes an ID Token previously issued by the OP to the client as a hint about the End-User's current or former session.
login_hint	OPTIONAL	*String*	This parameter passes a hint to the OpenID Provider (OP) regarding the login identifier (such as an email address or username) that the End-User might use to sign in. e.g: janedoe@example.com, janesmith42, +15555551234
acr_values	OPTIONAL	*String*	A space-delimited, case-sensitive list of ASCII strings that specifies the requested Authentication Context Class Reference (ACR) values. These represent the "strength" or specific methods of authentication. E.g: urn:mace:incommon:iap:silver, http://id.incommon.org/assurance/certification/uniquiz

Response Parameters

Parameters	Presence	Value	Description
code	MANDATORY	*String*	The authorization code generated by the authorization server. It shall be bound to the client identifier and the redirection URI. It shall expire shortly after it is issued to mitigate the risk of leaks. The signature application cannot use the value more than once.
state	CONDITIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.
error_uri	OPTIONAL	*String*	A URI identifying a human-readable web page with information about the error. It may be returned only in case of an error.

3.2.14 OAuth2/PushedAuthorize

It does not specify a regular CSC API method, but rather the URI path component of the address of the web page allowing the user to sign-in to the remote service to authorize the signature application or to

authorize a credential. The complete URL to invoke the OAuth 2.0 authorization server is obtained by adding oauth2/pushed_authorize to the base URI of the authorization server as returned in the oauth2 parameter by the “info” method and it does not necessarily include the base URI of the remote service API. The difference between oauth2/authorize and oauth2/pushed_authorize is that it return directly request URI to client application instead of redirecting on location header.

https://<server>:8778/adss/service/unity/csc/v2/oauth2/pushed_authorize	
HTTP Verb	POST
Content-Type	
Accept	
Parameters	**//Service Authorization** response_type=code& client_id=samples_test_client& redirect_uri =http://localhost:8777& scope=service& lang=en-UK& state=123456& code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U & code_challenge_method=S256& login_hint=janesmith42& acr_values=urn:mace:incommon:iap:silver& display=page& response_mode=fragment& id_token_hint=eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJpc3MiOiJodHRwczov& max_age=3600& profile_id=adss:unity:profile:001 **// Credentials Authorization** response_type=code& client_id=samples_test_client& redirect_uri =http://localhost:8777& scope=credential& credentialID=sample-key& numSignatures=2&

	hash= MTIzNDU2Nzg5MHF3ZXJ0enVpb3Bhc2RmZ2hqa2zDtnl4& state=12345 & code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U& code_challenge_method=S256& login_hint=janesmith42& acr_values=urn:mace:incommon:iap:silver& display=page& response_mode=fragment& id_token_hint=eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJpc3MiOiJodHRwczov& max_age=3600

Status Code	Message	Response Body

302	Found	{
		"request_uri":"urn:example:bwc4JKESC0w8acc191eY1
		LTC2", "expires_in": 90
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

400	Bad Request	{
		"error": "65281",
		"error_description": "Failed to process request - Parameter: %ARGUMENT% is Invalid."
		}

400	Bad Request	{
		"error": "65282",
		"error_description": "Failed to process request - Parameter: max_age must be a valid Integer"
		}

400	Bad Request	{
		"error": "65283",
		"error_description": "Failed to process request - Optional parameter values exceed the allowed limit of 2000."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Request Parameters

Parameters	Presence	Value	Description
response_type	MANDATORY	*String*	The value shall be “code”.
redirect_uri	OPTIONAL	*String*	The URL where the user will be redirected after the authorization process has completed. Only a valid URI pre-registered with the UNITY Service shall be passed. If omitted, the service will use the default redirect URI pre-registered by the signature application. **Note:** If the Allowed Redirect URI(s) list is configured, the default redirect URI specified in the Redirect URI field must also be added to this list for validation. If the default URI is not included in the allowed list, the authorization request will be rejected, and an error will be returned.
scope	OPTIONAL	*String*	The scopes of the access request are mentioned below: • **Service:** It shall be used to obtain an authorization code suitable for service authorization. • **Credential:** It shall be used to obtain an authorization code suitable for credentials authorization. The parameter is OPTIONAL. The defaults scope is “service” in case it is omitted.
authorization_details	OPTIONAL	*String*	The details of the access request as described in IETF Draft-ietf-oauth-rar [27]. This specification defines the following authorization details type: • “credential”: it SHALL be used to obtain an authorization code suitable for credentials authorization. The parameter is OPTIONAL. If this parameter is used, all values relevant for credential authorization SHALL be passed in this object. The scope “credential” as well as any request parameter relevant for credential authorization SHALL NOT be used in this case.
prompt	OPTIONAL	*String*	When scope=service, the OIDC prompt parameter can be used to specify whether the Identity Provider (IDP) should prompt the end-user for re-authentication or consent. The Unity Service forwards the prompt value to the IDP if it is present in the request. If not present, the default value prompt=login is forwarded to the IDP.

			When scope=credential, the prompt value is derived from the Unity Profile, and any value provided in the request is ignored. Valid prompt values include: • none • consent • login • select_account
code_challenge	MANDATORY	*String*	Cryptographic nonce binding the transaction to a certain user agent, used to detect code replay and CSRF attacks.
code_challenge_method	OPTIONAL	*String*	Code verifier transformation method as defined in IETF RFC7636 [25], defaults to plain. The recommended value is S256.
acceptLanguage	OPTIONAL	*String*	Request a preferred language according to RFC 5646. If a language is explicitly specified, the authorization server should display the authorization web page in that language, provided it is supported. If no language is specified but an Accept-Language header is included in the request, the authorization server should use the language indicated in the header, if supported. **Note:** The lang parameter in the OAuth 2.0 /authorize request allows business applications to specify the desired language for the authorization page, ensuring it appears in a supported language. If no language preference is provided, the authorization server will default to its pre-configured language setting.
state	OPTIONAL	*String*	Up to 255 bytes of arbitrary data from the signature application that will be passed back to the redirect URI. The use is recommended for preventing cross-site request forgery.
request_uri	CONDITIONAL	*String*	URI pointing to a pushed authorization request previously uploaded by the client. This parameter SHALL only be used in conjunction with the client_id. All other parameters SHALL NOT be combined with this parameter.
credentialID	CONDITIONAL	*String*	The identifier associated to the credential to authorize. It shall be used only if the scope of the OAuth 2.0 authorization request is “credential”. This parameter value may contain characters that are reserved, unsafe or forbidden in URLs and therefore shall be url-

			encoded by the signature application (max. 50 characters).
numSignatures	CONDITIONAL	*Number*	The number of signatures to authorize. Multi-signature transactions can be obtained by using a combination of array of hash values and by calling multiple times the **signatures/signHash** method. It shall be used only if the scope of the OAuth 2.0 authorization request is “credential”.
hashes	CONDITIONAL	*String*	One or more base64url-encoded hash values to be signed. It allows the server to bind the SAD to the hash, thus preventing an authorization to be used to sign a different content. It shall be used if the SCAL parameter returned by credentials/info method, for the current credentialID is “2”, otherwise it is OPTIONAL. Multiple hash values can be passed as comma separated values, e.g. oauth2/authorize?hash=dnN3ZX…ZmRm,ZjIxM3…Z2Zk,… The order of multiple values does not have to match the order of hashes passed to **signatures/signHash** method.
hashAlgorithmOID	CONDITIONAL	*String*	String containing the OID of the hash algorithm used to generate the hashes.
description	OPTIONAL	*String*	A free form description of the authorization transaction in the lang language. The maximum size of the string is 5000 characters. It can be useful to provide some hints about the occurring transaction.
account_token	CONDITIONAL	*String*	An account_token may be required by a RSSP if their authorization server has a restricted access. The value is a JSON Web Token (JWT) according to RFC 7519. It shall be passed if no client_id is used, particularly in cases when there are huge number of clients and each of them cannot be registered in the ADSS server, so instead of passing the client_id, an account_token can be passed. **Note:** The account_token can be generated via OAuth2/Token API by passing the client_assertion and client_assertion_type with grant_type as client_credentials.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.
response_mode	OPTIONAL	*String*	This parameter informs the OP of the mechanism to be used when returning authorization response parameters (like codes

			or tokens) from the Authorization Endpoint. E.g: query, fragment
display	OPTIONAL	*String*	An ASCII string value that provides a hint to the OP about how it should display the user interface pages (authentication and consent screens) to the End-User. E.g: page,wap
max_age	OPTIONAL	*Number*	Description: Specifies the allowable elapsed time (in seconds) since the last time the End-User was actively authenticated by the OpenID Provider. E.g: 0,3600
Id_token_hint	OPTIONAL	*String*	This parameter passes an ID Token previously issued by the OP to the client as a hint about the End-User's current or former session.
login_hint	OPTIONAL	*String*	This parameter passes a hint to the OpenID Provider (OP) regarding the login identifier (such as an email address or username) that the End-User might use to sign in. e.g: janedoe@example.com, janesmith42, +15555551234
acr_values	OPTIONAL	*String*	A space-delimited, case-sensitive list of ASCII strings that specifies the requested Authentication Context Class Reference (ACR) values. These represent the "strength" or specific methods of authentication. E.g: urn:mace:incommon:iap:silver, http://id.incommon.org/assurance/certification/ uniquiz

Response Parameters

Parameters	Presence	Value	Description
request_uri	MANDATORY	*String*	URI path component of the address of the web page allowing the user to sign-in to the remote service to authorize the signature application or to authorize a credential.
expires_in	OPTIONAL	*Number*	The lifetime in seconds of request_uri validity.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.
error_uri	OPTIONAL	*String*	A URI identifying a human-readable web page with information about the error. It may be returned only in case of an error.

Table 35 – OAuth2/PushedAuthorize

3.2.15 OAuth2/Token – Authorization Code Flow

Obtain an OAuth 2.0 bearer access token from the authorization server by passing the authorization code or refresh token returned by the authorization server after a successful user authentication, along with the client ID and client secret in possession of the signature application.

https://<server>:8778/adss/service/unity/csc/v1/oauth2/token		
HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Header		
profile_id	adss:unity:profile:001	
Request Body	grant_type=authorization_code& client_id=samples_test_client& client_secret=jr67gj0h76gr83nf8734nj59g4he895jh87nr& code=ssd34343& redirect_uri=http://locahost:8777& code_verifier=test	
Status Code	**Message**	**Response Body**

200	OK	// Service Authorisation Response
		{
		"access_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"refresh_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"token_type":"Bearer"
		"expires_in":"3600"
		}
		// Credentials Authorisation Response
		{
		"access_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"token_type":"SAD"
		"expires_in":"3600"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 36 – OAuth2/Token – Authorization Code Flow

Request Parameters

Parameters	Presence	Value	Description
grant_type	MANDATORY	*String*	The grant type, which depends on the type of OAuth 2.0 flow: • **authorization_code:** It shall be used in case of Authorization Code Grant. • **client_credentials:** It shall be used in case of Client Credentials Grant. • **refresh_token:** It shall be used in case of Refresh Token flow.
refresh_token	CONDITIONAL	*String*	The long-lived refresh token returned from the previous session. This shall be used only when the scope of the OAuth 2.0 authorization request is "service" and *grant_type* is "refresh_token" to reauthenticate the user according to the method described in RFC 6749.
client_id	CONDITIONAL	*String*	The unique ''client ID'' previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 50 characters).
client_secret	CONDITIONAL	*String*	This is the "client secret" previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 200 characters).
code	CONDITIONAL	*String*	The authorization code returned by the authorization server. It shall be bound to the client identifier and the redirection URI. This shall be used only when *grant_type* is "authorization_code".
client_assertion	CONDITIONAL	*String*	The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents. It shall be passed if no authorization header and no *client_secret* is used.
client_assertion_type	CONDITIONAL	*String*	The format of the assertion as defined by the authorization server. The value will be an absolute URI. It shall be passed if a client_assertion is used.
redirect_uri	CONDITIONAL	*String*	The URL where the user was redirected after the authorization process completed. It is used to validate that it matches the original value previously passed to the authorization server. This shall be used only if the redirect_uri parameter was included in the

			authorization request, and their values shall be identical.
profile_id	CONDITIONAL	*String*	UNITY Profile ID that will be used to process the request (max. 200 characters). It shall be passed if a client_assertion is used.
code_verifier	MANDATORY	*String*	A cryptographically random string that is used to correlate the authorization request to the token request as describe in RFC 7636

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	The short-lived access token to be used depending on the scope of the OAuth 2.0 authorization request. When the scope is “service” then the authorization server returns a bearer token to be used as the value of the “Authorization: Bearer” in the HTTP header of the subsequent API requests within the same session. When the scope is “credential” then the authorization server returns a Signature Activation Data token to authorize the signature request. This value should be used as the value for the SAD parameter when invoking the **signatures/signHash** method.
refresh_token	OPTIONAL	*String*	The long-lived refresh token used to re-authenticate the user on the subsequent session based on the method described in RFC 6749. The presence of this parameter is controlled by the user and is allowed only when the scope of the OAuth 2.0 authorization request is “service”. In case *grant_type* is “refresh_token” the authorization server may issue a new refresh token, in which case the client shall discard the old refresh token and replace it with the new refresh token.
token_type	MANDATORY	*String*	When the scope is “service”, this specifies a "Bearer" token type as defined in RFC6750 . When the scope is “credential”, this specifies a “SAD” token type.
expires_in	OPTIONAL	*Number*	The lifetime in seconds of the service access token. If omitted, the default expiration time is 3600 sec. (1 hour).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.16 OAuth2/Token – Client Credentials Flow

Obtain an OAuth 2.0 bearer access token from the authorization server by passing the client credentials which is pre-assigned by the authorization server to the signature application along with the client ID and client secret in possession of the signature application.

https://<server>:8778/adss/service/unity/csc/v1/oauth2/token		
HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Headers		
profile_id	adss:unity:profile:001	
Request Body	grant_type=client_credentials& client_id=samples_test_client& client_secret=jr67gj0h76gr83nf8734nj59g4he895jh87nr	
Status Code	**Message**	**Response Body**

200	OK	// Service Authorization Response
		{
		"access_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"refresh_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"token_type":"Bearer"
		"expires_in":"3600"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 37 – OAuth2/Token – Client Credentials Flow

Request Parameters

Parameters	Presence	Value	Description
grant_type	MANDATORY	*String*	The grant type, which depends on the type of OAuth 2.0 flow. In this case it will be "client_credentials".
client_id	CONDITIONAL	*String*	The unique ''client ID'' previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 50 characters).
client_secret	CONDITIONAL	*String*	This is the "client secret" previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 200 characters).
client_assertion	CONDITIONAL	*String*	The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents. It shall be passed if no authorization header and no *client_secret* is used.
client_assertion_type	CONDITIONAL	*String*	The format of the assertion as defined by the authorization server. The value will be an absolute URI. It shall be passed if a client_assertion is used.
profile_id	CONDITIONAL	*String*	UNITY Profile ID that will be used to process the request. It shall be passed if a client_assertion is used (max. 200 characters).

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	The short-lived access token to be used depending on the scope of the OAuth 2.0 authorization request.
token_type	MANDATORY	*String*	In case of client_credentials this will be "bearer".
expires_in	OPTIONAL	*Number*	The lifetime in seconds of the service access token. If omitted, the default expiration time is 3600 sec. (1 hour).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.17 OAuth2/Token – Refresh Token Flow

Obtain an OAuth 2.0 bearer access token from the authorization server by passing the client credentials with refresh token which is pre-assigned by the authorization server to the signature application along with the client ID and client secret in possession of the signature application.

https://<server>:8778/adss/service/unity/csc/v1/oauth2/token

HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Headers		
profile_id	adss:unity:profile:001	
Request Body	grant_type=refresh_token& refresh_token=Base64& client_id=samples_test_client& client_secret=jr67gj0h76gr83nf8734nj59g4he895jh87nr	
Status Code	**Message**	**Response Body**

200	OK	// Service Authorization Response
		{
		"access_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"refresh_token":"KeTob5gl26S2tmXjqN…MRGtoew=="
		"token_type":"Bearer"
		"expires_in":"3600"
		}

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 38 – OAuth2/Token – Refresh Token Flow

Request Parameters

Parameters	Presence	Value	Description
grant_type	MANDATORY	*String*	The grant type, which depends on the type of OAuth 2.0 flow. In this case it will be “refresh_token”.

client_id	CONDITIONAL	*String*	The unique ‘‘client ID’’ previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 50 characters).
client_secret	CONDITIONAL	*String*	This is the “client secret” previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 200 characters).
client_assertion	CONDITIONAL	*String*	The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents. It shall be passed if no authorization header and no *client_secret* is used.
client_assertion_type	CONDITIONAL	*String*	The format of the assertion as defined by the authorization server. The value will be an absolute URI. It shall be passed if a client_assertion is used.
refresh_token	MANDATORY	*String*	Existing refresh token used to get the new access and refresh tokens. **Note:** Existing refresh token is marked as revoked and cannot be used again if once used to get the new access and refresh tokens.
profile_id	CONDITIONAL	*String*	UNITY Profile ID that will be used to process the request. It shall be passed if a client_assertion is used (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	The short-lived access token to be used depending on the scope of the OAuth 2.0 authorization request.
refresh_token	OPTIONAL	*String*	The long-lived refresh token used to re-authenticate the user on the subsequent session based on the method described in RFC 6749. The presence of this parameter is controlled by the user and is allowed only when the scope of the OAuth 2.0 authorization request is “service”. In case *grant_type* is “refresh_token” the authorization server may issue a new refresh token, in which case the client shall discard the old refresh token and replace it with the new refresh token.
token_type	MANDATORY	*String*	In case of refresh_token this will be “bearer”.

expires_in	OPTIONAL	*Number*	The lifetime in seconds of the service access token. If omitted, the default expiration time is 3600 sec. (1 hour).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

3.2.18 OAuth2/Revoke

Revoke an access token or refresh token that was obtained from the authorization server, as described in RFC 7009. This method may be used to enforce the security of the remote service. When the signature application needs to terminate a session, it is RECOMMENDED to invoke this method to prevent further access by reusing the token.

https://<server>:8778/adss/service/unity/csc/v1/oauth2/revoke		
HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Body	token= jr67gj0h76gr83nf8734nj59g4he895jh87nr& token_type_hint=access_token/refresh_token& client_id=samples_test_client& client_secret=jr67gj0h76gr83nf8734nj59g4he895jh87nr	
Status Code	**Message**	**Response Body**
200	OK	

400	Bad Request	{
		"error": "invalid_request",
		"error_description": "The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."
		}

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 39 – OAuth2/Revoke

Request Parameters

Parameters	Presence	Value	Description

token	MANDATORY	*String*	The token that the signature application wants to get revoked.
token_type_hint	OPTIONAL	*String*	Specifies an optional hint about the type of the token submitted for revocation. If the parameter is omitted, the authorization server should try to identify the token across all the available tokens.
client_id	CONDITIONAL	*String*	The unique ‘‘client ID’’ previously assigned to the signature application by the remote service. It shall be passed if no authorization header is used. It shall be passed if no authorization header and no client_assertion is used (max. 50 characters).
client_secret	CONDITIONAL	*String*	This is the “client secret” previously assigned to the signature application by the remote service. It shall be passed if no authorization header and no client_assertion is used (max. 200 characters).
client_assertion	CONDITIONAL	*String*	The assertion being used to authenticate the client. Specific serialization of the assertion is defined by profile documents. It shall be passed if no authorization header and no *client_secret* is used. **Note:** If an access or refresh token to be revoked was generated via client_assertion, then it can also be revoked by passing the client_assertion and client_assertion_type instead of the client_id and client_secret.
client_assertion_type	CONDITIONAL	*String*	The format of the assertion as defined by the authorization server. The value will be an absolute URI. It shall be passed if a client_assertion is used.
clientData	OPTIONAL	*String*	Arbitrary data from the signature application. It can be used to handle a transaction identifier or other application-specific data that may be useful for debugging purposes. **Warning:** This parameter may expose sensitive data to the remote service. Therefore, it should be used carefully.
profile_id	CONDITIONAL	*String*	UNITY Profile ID that will be used to process the request. It shall be passed if a client_assertion is used (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4 Mobile Application Interfaces

A mobile app must interact with ADSS Unity Service to handle these services:

- Registration of the user's mobile device for remote authorisation
- Allowing the user to receive, authorise and send remote signing requests/responses

Mobile apps integrate with ADSS Unity Service using RESTful APIs. This section details each API method.

4.1 Authenticate Client

This API is used to authenticate a client using its credentials. The Unity Service returns an access token on successful authentication of the client.

https://<server>:8778/adss/service/unity/v1/authenticate		
HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Body	client_id=samples_test_client & client_secret=121212 & grant_type=client_credentials	
Status Code	**Message**	**Response Body**
200	OK	{ "access_token":"2YotnFZFEjr1zCsicMWpAA", "expires_in":3600 }
400	Bad Request	For Error information in client credentials request refer OAuth RFC 6749 at: https://tools.ietf.org/html/rfc6749#section-5.2
429	Too Many Requests	{ "error": "58129", "error_description": "Failed to process request - You have exhausted your API Request Quota" }

Table 40 – Authenticate Client

Request Parameters

Parameters	Presence	Value	Description
client_id	MANDATORY	*String*	Client ID registered in ADSS Client Manager (max. 50 characters).
client_secret	MANDATORY	*String*	Client Secret generated against the client in ADSS Client Manager (max. 200 characters).
grant_type	MANDATORY	*String*	The grant type, which depends on the type of OAuth 2.0 flow:

			• **client_credentials:** It shall be used in case of Client Credentials Grant. • **refresh_token:** It shall be used in case of Refresh Token flow. • **password:** It shall be used in case of password.
username	CONDITIONAL	*String*	If grant type is "password" then username request parameter is required and User name as friendly name for the registered user in Unity service (max. 50 characters).
password	CONDITIONAL	*String*	If grant type is "password" then password request parameter is required and provide the OTP received by the user's registered mobile number or user registered email.

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	It will return the client access token after the authentication of client ID and secret.
expires_in	MANDATORY	*String*	Token expiry mentioned in the seconds.

4.2 Authenticate User

This call initiates the user authentication on the mobile application. A user can be authenticated using the following authentication methods. These methods can be configured in the Unity Service Profile:

- Authenticate user with OTP(s) (Either SMS or Email or Both SMS/Email)
- Authenticate user with QR Code
- No Authentication

Authenticate user with OTP(s):

If this option is enabled, it means user will be authenticated using the OTPs. UNITY will send a request to SAM to generate either a single or two OTPs according to the option "SMS OTP" and "Email OTP" selected in the UNITY Profile. The SAM will generate the OTP(s) and return to UNITY that will send the OTP(s) to user's mobile number or email. Unity Service will return the authentication type "OTP" in response to mobile application to let it know that the user will be authenticated using OTP(s) that are sent to his/her mobile/email. The mobile application should display fields to user to enter the OTP(s) and once user enters the OTP, the mobile application will invoke another Unity Service API (Verify OTPs) to verify these OTP(s). The API is discussed in a later section.

Authenticate user with QR Code:

If this option will be selected in Unity Service Profile, the Unity Service will instantly return the response to mobile application with authentication type "qrCode". This will be an indication that the user will be authenticated using a QR Code so the mobile app will ask the user to go to QR code page and scan the QR code. Once the mobile app scans the QR Code, it will send this to UNITY for verification by calling another API (Verify QR Code).

No Authentication:

In this case, the Unity Service will just verify the user a registered one in the SAM Service. After getting confirmation from SAM the Unity Service will generate the access and refresh tokens for this user and return to client application. The presence of access token in response will be an indication for mobile app that the user has been authenticated.

Note: The clients should use “No Authentication” option in scenarios where the mobile app has its own mechanism of user authentication. The mobile app will authenticate the user and then request for an access token from UNITY to access its APIs.

https://<server>:8778/adss/service/unity/v1/user/enrol		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {application_access_token}	

Request Body	{
	"user_id": "John_Doe",
	}	

Response Header		
authentication_methods	true	
Status Code	**Message**	**Response Body**

200	OK	If OTP Authentication is configured in Unity Service Profile.
		
		If both SMS and Email OTPs are sent to user:
		{
		"auth_type": "OTP",
		"otp_info": [
		{
		"otp_type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		},
		{
		"otp_type": "SMS_OTP",
		"sent_to": "+448007720442"
		}
		]
		}

		If one OTP will be sent on user email:
		{
		"auth_type": "OTP",
		"otp_info": [
		{
		"otp_type": "EMAIL_OTP",
		"sent_to": "john.doe@sample.som",
		}
		]
		}

		If one OTP will be sent to user’s mobile: { "auth_type": "OTP", "otp_info": [{

		"otp_type": "SMS_OTP", "sent_to": "+448007720442" }] }

		If QR Code authentication is configured:
		{
		"auth_type": "QR_CODE",
		}

		If no authentication is configured:
		{
		"auth_type": "NO_AUTHENTICATION",
		"token_info": {
		"access_token":"eyJhbGciOiJIUzIda……96RDo",
		"refresh_token":"eyJhbGciOiJIUzsac……ymjGp-E",
		"token_type":"bearer",
		"expires_in":3600
		}
		}

400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 41 – Authenticate User

Request Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID as registered by the business application in ADSS Server SAM (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
auth_type	MANDATORY	*String*	Authentication type configured in UNITY Profile. The following values can be found in this parameter: - OTP - QR_CODE - NO_AUTHENTICATION **Note:** The values OTP, QR_CODE and NO_AUTHENTICATION are case-sensitive.

otp_info	CONDITIONAL	*String*	Contains information related to the types of OTPs (Email/SMS) and the mobile number and email of the user. It applies when OTP authentication is enabled in Unity Service Profile.
token_info	CONDITIONAL	*String*	Contains the OAuth access & refresh tokens and the expiry. It applies when "No Authentication" option is enabled in Unity Service Profile.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.3 Verify OTPs

If the OTP authentication will be enabled in Unity Service, the user will receive either one or two OTPs on his mobile number or email. The user will provide these OTPs to this API. After successful OTPs verification, access and refresh tokens are returned.

https://server:8778/adss/service/unity/v1/authentication/otp/verify	
HTTP Verb	POST
Content-Type	application/json
Accept	application/json
Authorization	Bearer {application_access_token}

Request Body	If user had received two OTPs:
	{
	"user_id": "User ID",
	"otp_info": [{
	"otp": "258456987",
	"otp_type": "SMS_OTP"
	},{
	"otp": "258456987",
	"otp_type": "EMAIL_OTP"
	}]
	}

	If user had received a single OTP on mobile: { "user_id": "User ID",

	"otp_info": [{
	"otp": "258456987",
	"otp_type": "SMS_OTP"
	}]
	}	

	If user had received one OTP via email:
	{
	"user_id": "User ID",
	"otp_info": [{
	"otp": "258456987",
	"otp_type": "EMAIL_OTP"
	}]
	}	

Status Code	**Message**	**Response Body**

200	Ok	{
		"access_token":"eyJhbGciOiJIUzI1NiIsInR5……EFRJ96RDo",
		"refresh_token":"eyJhbGciOiJIUzI1NiIsInR5……heyXmjGp-E",
		"token_type":"bearer",
		"expires_in":3600,
		}

400	Bad Request	
401	Unauthorized	
403	Forbidden	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 42 – Verify OTPs

Request Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID as registered by the business application (max. 50 characters).
otp_info	MANDATORY	*JSON Object*	It contains the OTPs and their types i.e. SMS/Email.

otp	MANDATORY	*String*	OTP received by the user on his/her mobile/email (max. 100 characters).
otp_type	MANDATORY	*String*	Type of the OTP i.e. SMS or Email (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	Access token of the user to use in subsequent API calls.
refresh_token	MANDATORY	*String*	Refresh token will be used to get the new access token without authenticating the user again.
expires_in	MANDATORY	*String*	It's token expiry mentioned in the seconds.
error	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.4 Renew Access Token

This API allows the renewal of an expired access token by providing the refresh token.

https://<server>:8778/adss/service/unity/v1/authenticate		
HTTP Verb	POST	
Content-Type	application/x-www-form-urlencoded	
Accept	application/json	
Request Body	grant_type=refresh_token&refresh_token=tGzv3JOkF0XG5Qx	
Status Code	**Message**	**Response Body**
200	OK	{ "access_token":"2YotnFZFEjr1zCsicMWpAA", "refresh_token":"TRVFHTHcedfJGJFLGKKJ", "expires_in":3600, }
400	Bad Request	For Error information in client credentials request refer OAuth RFC 6749 at: https://tools.ietf.org/html/rfc6749#section-5.2
429	Too Many Requests	{ "error": "58129", "error_description": "Failed to process request - You have exhausted your API Request Quota" }

Table 43 – Renew Access Token

Request Parameters

Parameters	Presence	Value	Description
grant_type	MANDATORY	*String*	Grant type would be refresh_token.
refresh_token	MANDATORY	*String*	Refresh token which mobile application already received after user authentication.

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	New access token.
refresh_token	MANDATORY	*String*	New refresh token to cover in-activity time by the logged-in user.
expires_in	MANDATORY	*String*	Access token expiry in seconds.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error message description

4.5 Device Registration

Once the mobile application gets the access token it can use other APIs of Unity Service. This API is used to register user's mobile device for remote signature authorisation purposes and request a certificate for the device's authorisation public key. Mobile application first needs to generate the key-pair in mobile device's software and hardware (Secure Enclave) and also generate the CSR (Certificate Signing Request). Once the CSR is generated, it will be sent in this API along with other information of the device. The UNITY Service will return the certificate generated for the device after registering the device.

https://<server>:8778/adss/service/unity/v1/authorization/certificate		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {access_token}	
Request Body	{ "csr": "MIICxDCCAawCAQAwfzELM[....]5f52oQ==", "device": { "device_id": " ASJMMN5389FF ", "device_name": "IPHONE X", "secure_element": true, "biometric": true, } }	
Status Code	**Message**	**Response Body**
200	OK	{ "alias": "hvcNAU+qCdXzADEA" "certificate": "MIIItAYJKocNA[...]ZU+qCdXzADEA" }
400	Bad Request	

500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 44 – Device Registration

Request Parameters

Parameters	Presence	Value	Description
csr	MANDATORY	*String*	Base64 encoded value of the CSR. A certificate will be issued for mobile device against this CSR.
device_name	MANDATORY	*String*	Alias of the device. Later can be renamed (max. 100 characters).
device_id	MANDATORY	*String*	A unique device ID to identify the device, For example, UUID random number (max. 255 characters).
secure_element	MANDATORY	*Boolean*	Must set to "True" if device has a hardware Secure Element/Enclave.
biometric	MANDATORY	*Boolean*	Must set to "True" if device has biometric feature available. It can be TouchID, FaceID, Fingerprint etc.

Response Parameters

Parameters	Presence	Value	Description
certificate	MANDATORY	*String*	Certificate generated for the device in base64 encoded format.
alias	MANDATORY	*String*	The certificate alias assigned by UNITY Service to this certificate (max. 255 characters).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

4.6 List Registered Devices – User Access Token

This API returns all the devices of a user that user has registered for use in remote authorised signing operations.

https://<server>:8778/adss/service/unity/v1/authorization/devices	
HTTP Verb	GET
Accept	application/json

Authorization	Bearer {user_access_token}	
Status Code	**Message**	**Response Body**

200	OK	[
		{
		"device_id": "id-001",
		"device_name": "IPhone",
		"secure_element": true,
		"biometric": true,
		"device_certificate": " MIIItAYJKocNA[...]ZU+qCdADEA "
		},
		{
		"device_id": "id-002",
		"device_name": "Samsung",
		"secure_element": true,
		"biometric": true,,
		"device_certificate": " MIIItAYJKocNA[...]ZU+qCdADEA "
		}
		]

400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 6 – List Registered Devices – User Access Token

Response Parameters

Parameters	Presence	Value	Description
device_id	MANDATORY	*String*	Device ID (max. 255 characters).
device_name	MANDATORY	*String*	Device alias (max. 255 characters).
device_certificate	MANDATORY	*String*	Certificate generated for the device in base64 encoded format.
secure_element	MANDATORY	*String*	“True” if device has secure element/enclave.
biometric	MANDATORY	*String*	“True” if device has biometric feature available.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message

4.7 List Registered Devices – Client Credentials

This API returns all the devices of a user that user has registered for use in remote authorised signing operations.

https://<server>:8778/adss/service/unity/v1/authorization/devices?user_id={user_id}

HTTP Verb	GET
Accept	application/json
Authorization	Bearer {application_access_token}
Request Body	

Status Code	Message	Response Body

200	OK	[
		{
		"device_id": "id-001",
		"device_name": "IPhone",
		"secure_element": true,
		"biometric": true,
		"device_certificate": " MIIItAYJKocNA[...]ZU+qCdADEA "
		},
		{
		"device_id": "id-002",
		"device_name": "Samsung",
		"secure_element": true,
		"biometric": true,
		"device_certificate": " MIIItAYJKocNA[...]ZU+qCdADEA ",
		}
		]

400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 7 – List Registered Devices – Client Credentials

Request Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID whose devices information needs to be retrieved (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
device_id	MANDATORY	*String*	Device ID (max. 255 characters).
device_name	MANDATORY	*String*	Device alias (max. 255 characters).
device_certificate	MANDATORY	*String*	Certificate generated for the device in base64 encoded format.

secure_element	MANDATORY	*String*	"True" if device has secure element/enclave.
biometric	MANDATORY	*String*	"True" if device has biometric feature available.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message

Note: The List Registered Devices (Client Credentials) API is deprecated due to user privacy concerns, hence it will removed in future versions of the ADSS Server. Now this API could be used in a restricted manner. By default, it will not work with Client Credentials. Therefore, to use this API with Client Credentials, we must set the property "MOBILE_API_AUTHENTICATION" value to TRUE. If you want more details, please refer to the link here.

4.8 Delete Device

This API deletes a user's device in Unity Service identified by {device_id}. A client application would use this interface to delete a user's device.

https://server:8778/adss/service/unity/v1/authorization/devices/{device_id}		
HTTP Verb	DELETE	
Accept	application/json	
Access Token	Bearer {user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**
200	OK	
404	Not Found	
403	Forbidden	
500	Internal Server Error	
429	Too Many Requests	

Table 45 – Delete Device

Request Parameters

Parameters	Presence	Value	Description
{device_id}	MANDATORY	*String*	Device ID which is already registered in ADSS Server (max. 255 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.9 Get a Pending Authorisation Request

This method returns a pending authorisation request. That is, where the business application has requested a signing operation that requires user authorisation. It will return only a single request to process by the client application.

https://<server>:8778/adss/service/unity/v1/authorization/request		
HTTP Verb	GET	
Accept	application/json	
Authorization	Bearer {user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	{
		"transaction_id": "932469001521668267",
		"request": "PEFDRj48Y2VydEFs[...]9BQ0Y+",
		"hash_algorithm": “SHA256”,
		"display_text": “Authorisation Required”,
		"sad_format": “XML/JSON”,
		"authorization_data": {
		"transaction_id": " 932469001521668267",
		"originator_id": "samples_test_client",
		"user_id": "John Doe",
		"certificate_id": "sample_cert_alias",
		"salt": "651589814845846999",
		"device_id": "",
		"number_signatures": "1",
		"documents": [{
		"document_id": 123,
		"document_name": “Document Name 123”,
		}, {
		"document_id": 456,
		"document_name": “Document Name 456”,
		}],
		"validity_period": {
		"valid_from": "2023-08-08T11:01:19",
		"valid_to": "2023-08-08T19:01:19"
		}
		}
		}

400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 46 – Get a Pending Authorisation Request

Response Parameters

Parameters	Presence	Value	Description
transaction_id	MANDATORY	*String*	Transaction ID that uniquely identifies the request (max. 100 characters).
request	CONDITIONAL	*String*	Pending authorization request in base64 form - this is the "object" together with some additional data e.g. Device ID added and signed by the mobile app using the authorisation key. Value must be decoded before signing operation. After decoding the request, it will be look like this. This field must be present in case of XML as the sad_format. <AuthorisationData> <OriginatorID>Virtual_CSP_Client</OriginatorID> <UserID>olcayatli@gmail.com</UserID> <CertificateID>416edc72-6c63-45aa-bb34a373102234df</CertificateID> <TransactionID>980551837300673581</TransactionID> <Salt>92455249529156563 2</Salt> <MetaData> <DisplayText>Data to be displayed</DisplayText> <DeviceID></DeviceID> </MetaData> <Documents> <Document id="b81e040a-a4d8-4134-92ff-2d4bf5e9116d"> <Name>b81e040a-a4d8-4134-92ff-2d4bf5e9116d</Name> <DigestValue>ypkP9L2tZO2JdfNr4X4X5SRur529uJqykdc5q5HDSiLNiYcLrysO0S/H31yb8QZS
SOBYsFlVSj9/SKUqrhsUC5oEc/gr</DigestValue> </Document> </Documents> <ValidityPeriod> <ValidFrom>2019-12-07T18:25:37</ValidFrom> <ValidTo>2019-12-07T18:42:17</ValidTo> </ValidityPeriod> <Signature> <DigestMethod>SHA256</DigestMethod> </AuthorisationData>
hash_algorithm	MANDATORY	*String*	Hash algorithm to be used for signing the authorized remote signing Request (max. 500 characters).
display_text	CONDITIONAL	*String*	Message to be shown on the mobile device. This field must be present in case of JSON as the sad_format.
sad_format	MANDATORY	*String*	The format of Signature Activation Data. Possible values are JSON and XML.
authorization_data	CONDITIONAL	*String*	JSON object containing SAD. This field must be present in case of JSON as the sad_format.
transaction_id	MANDATORY	*String*	Unique identifier of the request.

originator_id	MANDATORY	*String*	Client's Originator ID which is configured in ADSS Console > Client Manager
user_id	MANDATORY	*String*	User ID identifying the registered user in SAM service.
certificate_id	MANDATORY	*String*	Certificate ID identifying the certificate to sign the hash.
salt	MANDATORY	*String*	Salt to be used while performing authorisation.
device_id	CONDITIONAL	*String*	Unique ID for the identification of the device.
number_signatures	MANDATORY	*String*	Number of signatures user needs to provide authorisation for.
documents	MANDATORY	*String*	JSON Array containing documents to be signed.
document_id	MANDATORY	*String*	ID of the document to be signed.
document_name	MANDATORY	*String*	Name of the document to be signed.
digest_value	MANDATORY	*String*	Hash of document to be signed.
validity_period	MANDATORY	*String*	JSON Object containing validity related information.
valid_from	MANDATORY	*String*	Time at which the SAD is issued.
valid_to	MANDATORY	*String*	Time till which the SAD is valid.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.10 Get Pending Authorisation Requests

This method returns the list of the pending authorisation request. That is, where the business application has requested multiple signing operations that requires user authorisation. It will return the list of requests to process by the client application.

https://<server>:8778/adss/service/unity/v1/authorization/request/list?{query_param}		
HTTP Verb	GET	
Accept	application/json	
Authorization	Bearer {user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	[
		{
		"transaction_id": "961381365804386927",
		"request": "PEF1dGhvcmlzYXRpb25EYXRhPjxPcmlnaW5hdG9ySUQ=QQC",
		"hash_algorithm": "SHA256" ,
		"display_text": "Authorisation Required",
		"sad_format": "XML/JSON",
		"authorization_data": {
		"transaction_id": "961381365804386927",
		"originator_id": "samples_test_client",
		"user_id": "John Doe",
		"certificate_id": "sample_cert_alias",
		"salt": "651589814845846999",
		"device_id": "",

```
            "number_signatures": "1",
            "documents": [{
                "document_id": " document_id_01",
                "document_name": "Sample Document",
                "digest_value": "ATsCAQAwDjE="
            }],
            "validity_period": {
                "valid_from": "2023-08-08T11:01:19",
                "valid_to": "2023-08-08T19:01:19"
            }
        }
    },
    {
        "transaction_id": "734535860865120121",
        "request": "PEF1dGhvcmlzYXRpb25EYXRhPjxPcmlnaW5
hdG9ySUQ=QQCSDFASADSDSA",
        "hash_algorithm": "SHA256",
        "display_text": "Authorisation Required",
        "sad_format": "XML/JSON",
        "authorization_data": {
            "transaction_id": "734535860865120121",
            "originator_id": "samples_test_client",
            "user_id": "John Doe",
            "certificate_id": "sample_cert_alias",
            "salt": "651589814845846989",
            "device_id": "",
            "number_signatures": "1",
            "documents": [{
                "document_id": " document_id_01",
                "document_name": "Sample Document",
                "digest_value": "ATsCAQAwDjE="
            }],
            "validity_period": {
                "valid_from": "2023-08-08T11:01:19",
                "valid_to": "2023-08-08T19:01:19"
            }
        }
    },
    {
        "transaction_id": "626530850567912803",
        "request": "PEF1dGhvcmlzYXRpb25EYXRhPjxPcmlnaW5
hdG9ySUQ=QQCSDASJKDADJLSDAJADSLJKASDA",
        "hash_algorithm": "SHA256",
        "display_text": "Authorisation Required",
        "sad_format": "XML/JSON",
        "authorization_data": {
            "transaction_id": "626530850567912803",
            "originator_id": "samples_test_client",
            "user_id": "John Doe",
            "certificate_id": "sample_cert_alias",
            "salt": "6515898148458469899",
            "device_id": "",
```

		"number_signatures": "1",
		"documents": [{
		"document_id": " document_id_01",
		"document_name": "Sample Document",
		"digest_value": "ATsCAQAwDjE="
		}],
		"validity_period": {
		"valid_from": "2023-08-08T11:01:19",
		"valid_to": "2023-08-08T19:01:19"
		}
		}
		},
		{
		"transaction_id": "702563194319561527",
		"request": "PEF1dGhvcmlzYXRpb25EYXRhPjxSDSA",
		"hash_algorithm": "SHA256",
		"display_text": "Authorisation Required",
		"sad_format": "XML/JSON",
		"authorization_data": {
		"transaction_id": "702563194319561527",
		"originator_id": "samples_test_client",
		"user_id": "John Doe",
		"certificate_id": "sample_cert_alias",
		"salt": "651589814845845999",
		"device_id": "",
		"number_signatures": "1",
		"documents": [{
		"document_id": " document_id_01",
		"document_name": "Sample Document",
		"digest_value": "ATsCAQAwDjE="
		}],
		"validity_period": {
		"valid_from": "2023-08-08T11:01:19",
		"valid_to": "2023-08-08T19:01:19"
		}
		}
		}
		]

400	Bad Request	{
		"error": "58130",
		"error_description": "Failed to process request - Order is invalid in the request"
		}

500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 47 – Get Pending Authorisation Requests

Request Parameters

Parameters	Presence	Value	Description
{query_param}	OPTIONAL	*String*	Currently supported parameters are: • Order • startIndex • fetchSize Response will contain the array of the pending authorization request eg /adss/service/unity/v1/authorization/request/list?startIndex=2&fetchSize=2&order=desc **startIndex**: It indicates the index or position from which the API should start fetching the data. **fetchSize**: It determines the number of items or records the API should fetch in the response. **order**: It indicates the order in which the API should return the data. **Note:** Maximum number of records which can be returned by this API are 100. Therefore, if fetchSize is not provided in the request or is greater than 100, then by default 100 records would be returned by the API.

Response Parameters

Parameters	Presence	Value	Description
transaction_id	MANDATORY	*String*	Transaction ID that uniquely identifies the request (max. 100 characters).
request	CONDITIONAL	*String*	Pending authorization requests are in base64 form - this is the "object" together with some additional data e.g. Device ID added and signed by the mobile app using the authorization key. Value must be decoded before signing operation. After decoding the request, it will be look like this. This field must be present in case of XML as the sad_format. <AuthorisationData> <OriginatorID>Virtual_CSP_Client</OriginatorID> <UserID>olcayatli@gmail.com</UserID> <CertificateID>416edc72-6c63-45aa-bb34a373102234df</CertificateID> <TransactionID>980551837300673581</TransactionID> <Salt>924552495291565632</Salt> <MetaData> <DisplayText>Data to be displayed</DisplayText> <DeviceID></DeviceID> </MetaData> <Documents> <Document id="b81e040a-a4d8-4134-92ff-2d4bf5e9116d">

			<Name>b81e040a-a4d8-4134-92ff-2d4bf5e9116d</Name> <DigestValue>ypkP9L2tZO2JdfNr4X4X5SRur529uJqykdc5q5HDSiLNiYcLrysO0S/H31yb8QZS
SOBYsFlVSj9/SKUqrhsUC5oEc/gr</DigestValue> </Document> </Documents> <ValidityPeriod> <ValidFrom>2019-12-07T18:25:37</ValidFrom> <ValidTo>2019-12-07T18:42:17</ValidTo> </ValidityPeriod> <Signature> <DigestMethod>SHA256</DigestMethod> </AuthorisationData>
hash_algorithm	MANDATORY	*String*	Hash algorithm to be used for signing the authorized remote signing. Request (max. 500 characters).
display_text	CONDITIONAL	*String*	Message to be shown on the mobile device. This field must be present in case of JSON as the sad_format.
sad_format	MANDATORY	*String*	The format of Signature Activation Data. Possible values are JSON and XML.
authorization_data	CONDITIONAL	*String*	JSON object containing SAD. This field must be present in case of JSON as the sad_format.
transaction_id	MANDATORY	*String*	Unique identifier of the request.
originator_id	MANDATORY	*String*	Client's Originator ID which is configured in ADSS Console > Client Manager
user_id	MANDATORY	*String*	User ID identifying the registered user in SAM service.
certificate_id	MANDATORY	*String*	Certificate ID identifying the certificate to sign the hash.
salt	MANDATORY	*String*	Salt to be used while performing authorisation.
device_id	CONDITIONAL	*String*	Unique ID for the identification of the device.
number_signatures	MANDATORY	*String*	Number of signatures user needs to provide authorisation for.
documents	MANDATORY	*String*	JSON Array containing documents to be signed.
document_id	MANDATORY	*String*	ID of the document to be signed.
document_name	MANDATORY	*String*	Name of the document to be signed.
digest_value	MANDATORY	*String*	Hash of document to be signed.
validity_period	MANDATORY	*String*	JSON Object containing validity related information.
valid_from	MANDATORY	*String*	Time at which the SAD is issued.
valid_to	MANDATORY	*String*	Time till which the SAD is valid.
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.11 Authorise a Pending Request

This method authorises a pending request by sending the signed Signature Activation Data (SAD) against the pending authorisation request received as described above. That is, the value returned in section 4.9 above (together with some additional data) must be signed on the mobile device and returned using this

API. The returned value must be base64 encoded. The hash algorithm is as returned in section 4.9 above, and the same value is returned here in the body request.

https://<server>:8778/adss/service/unity/v1/authorization/request/{request_id}		
HTTP Verb	PUT	
Authorization	Bearer {user_access_token}	
Content-Type	application/json	
Accept	application/json	

Request Body	{
	"request": "PEFDRj48Y2VydEFs[...]9BQ0Y+",
	“hash_algorithm": “SHA256”
	}	

Status Code	**Message**	**Response Body**
200	OK	
400	Bad request	
500	Internal Server Error	
429	Too Many Requests	

Table 48 – Authorise a Pending Request

Request Parameters

Parameters	Presence	Value	Description
{request_id}	MANDATORY	*String*	Request ID received in section 4.9 above
Request	MANDATORY	*String*	In case of XML as the sad_format: Mobile app needs to decode the request, add <DeviceID> and then sign with private key and add the signature in the request. This will be sent in request in Base64 encoded format (max. 100 characters). This signed request also called the SAD (Signature Activation Data) will look like this: <AuthorisationData> <OriginatorID>Virtual_CSP_Client</OriginatorID> <UserID>olcayatli@gmail.com</UserID> <CertificateID>416edc72-6c63-45aa-bb34a373102234df</CertificateID> <TransactionID>980551837300673581</TransactionID> <Salt>92455249529156563 2</Salt> <MetaData>

<DisplayText>Data to be displayed</DisplayText>
<DeviceID>ad1e60a6-5e23-4b52-a127-27f41c224c05</DeviceID>
</MetaData>
<Documents>
<Document id="b81e040a-a4d8-4134-92ff-2d4bf5e9116d">
<Name>b81e040a-a4d8-4134-92ff-2d4bf5e9116d</Name>

<DigestValue>ypkP9L2tZO2JdfNr4X4X5SRur529uJqykdc5q5HDSiLNiYcLrysO0S/H31yb8QZS
SOBYsFlVSj9/SKUqrhsUC5oEc/gr</DigestValue>
</Document>
</Documents>
<ValidityPeriod>
<ValidFrom>2019-12-07T18:25:37</ValidFrom>
<ValidTo>2019-12-07T18:42:17</ValidTo>
</ValidityPeriod>
<Signature>
<DigestMethod>SHA256</DigestMethod>

<SignatureValue>MEUCID/kiJWAIqzwOp/hi+FUbJwsjdcsEoBNwlIF8sXA8XbDAiEAkGgQomyoJ2iR0ra9KGBFW/zXi6tbsn5M49YiaPNc+L8=</SignatureValue>
</AuthorisationData>

In case of JSON as the sad_format:

Mobile app must set the device_id field in authorization_data object then get it signed with private key to get the JSON Web Signature in the following format.

JSON Web Signature:

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NksifQ.eyJ0cmFuc2FjdGlvbl9pZCI6Ijg5MDAxMTM1MzQ3Nzk5NzQyNyIsIm9yaWdpbmF0b3JfaWQiOiJzYW1wbGVzX3RlY3RfY2xpZW50IiwidXNlcl9pZCI6ImFtMiIsImNlcnRpZmljYXRlX2lkIjoiYW0yLWtleSIsInNhbHQiOiI3MzU3NzU4NzUxMDk1NDU2MTciLCJkZXZpY2VfaWQiOiIiLCJudW1iZXJfc2lnbmF0dXJlcyI6IjIiLCJkb2N1bWVudHMiOlt7ImRvY3VtZW50X25hbWUiOiIiLCJkaWdlc3RfdmFsdWUiOiIyNDY3YzI5MTA3OGI0ZDFkY2Y3ODllYmUyNGIxMGJhNTBhZGQxMTk1MDQ4ZGU1YjA2MGZlMjlhMTczOThlZGE3In1dLCJ2YWxpZGl0eV9wZXJpb2QiOnsidmFsaWRfZnJvbSI6IjIwMjMtMDgtMDdUMTc6NTI6NTMiLCJ2YWxpZF90byI6IjIwMjMtMDgtMDhUMDE6NTE6NDMifX0.Yf-b-GbYnz1689b7iqHwkRNeOvCpg_aW7_AGJ_I5TySkoOnu2Ba64_qBhs6BAN-kfQ7hJzfIdZal4j6J3oA5Qw

The above JWS is computed against the below header and payload.

			Header:
			{
			"typ": "JWT",
			"alg": "ES256K"
			}
			Payload:
			{
			"transaction_id": "890011353477997427",
			"originator_id": "samples_test_client",
			"user_id": "am2",
			"certificate_id": "am2-key",
			"salt": "735775875109545617",
			"device_id": "",
			"number_signatures": "2",
			"documents": [
			{
			"document_name": "",
			"digest_value": "2467c291078b4d1dcf789ebe24b10ba50add1195048de5b060fe29a17398eda7"
			}
			],
			"validity_period": {
			"valid_from": "2023-08-07T17:52:53",
			"valid_to": "2023-08-08T01:51:43"
			}
			}

hash_algorithm	MANDATORY	*String*	Hashing algorithm used for signing SAD (max. 50 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.12 Cancel a Pending Authorisation Request

This method cancels a pending authorisation request. That is, the user decides to decline the authorisation request sent to the mobile device.

https://<server>:8778/adss/service/unity/v1/authorization/request/{request_id}	
HTTP Verb	DELETE
Authorization	Bearer {application_access_token | user_access_token }
Accept	application/json
Request Body	

Status Code	Message	Response Body
200	OK	
400	Bad request	
500	Internal Server Error	
429	Too Many Requests	

Table 49 – Cancel a Pending Authorisation Request

Request Parameters

Parameters	Presence	Value	Description
{request_id}	MANDATORY	*String*	Request ID for which user cancelled the authorization (max. 100 characters).

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.13 Users Profile

This API is used to get user's profile information from ADSS.

https://<server>:8778/adss/service/unity/v1/users/profile		
HTTP Verb	GET	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {user_access_token}	
Status Code	**Message**	**Response Body**
200	OK	{ "user_id": "Alice", "user_name": "Alice", "app_name": "samples_test_client", "user_email": "abc@ascertia.com", "user_mobile": "+9230XXXXXXXX" }
400	Bad Request	
500	Internal Server Error	
429	Too Many Requests	{ "error": "58129", "error_description": "Failed to process request - You have exhausted your API Request Quota" }

Table 50 – Users Profile

Response Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID that identifies the user (max. 50 characters).
user_name	MANDATORY	*String*	It's the user name (max. 200 characters).
App_name	CONDITIONAL	*String*	Application name the user belongs to (max. 50 characters).
user_email	MANDATORY	*String*	User email (max. 500 characters).
user_mobile	MANDATORY	*String*	User mobile number (max. 100 characters).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	A string with the description of the error_code.

4.14 Get Device Registration Settings

This interface is used to get the device related settings configured in the Unity Service Profile for the devices of a user. This API should be invoked before registering a device to check which key length & key type will be used to generate an authorisation key-pair and other settings e.g. where to generate this key-pair i.e. in Device Secure Enclave or Software KeyStore/KeyChain.

http://server:8778/adss/service/unity/v1/users/profile/device/settings		
HTTP Verb	GET	
Accept	application/json	
Authorization	Bearer {user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	{
		"device_key_type": "ECDSA",
		"device_key_size": 256,
		"secure_element_required": true,
		"biometric_required": true,
		"allowed_devices": 10,
		"clock_tolerance_on_auth_cert": 10
		
		}

400	Bad Request	
403	Forbidden	
404	Not Found	
500	Internal Server Error	
429	Too Many Requests	{ "error": "58129", "error_description": "Failed to process request - You have exhausted your API Request Quota"

		}

Table 51 – Get Device Registration Settings

Response Parameters

Parameters	Presence	Value	Description
device_key_type	MANDATORY	*String*	Key pair type to be generated in mobile device e.g. RSA Possible values are RSA & ECDSA (max. 500 characters).
device_key_size	MANDATORY	*Integer*	Key pair size e.g. 2048
secure_element_required	MANDATORY	*String*	If set "TRUE" then the authorisation key pair must be generated inside device secure enclave otherwise software key store or KeyChain be used for key pair generation. If this flag is set to TRUE and the device does not support a Secure Element, then an error should be generated by mobile app.
biometric_required	MANDATORY	*String*	If set TRUE then device must have biometric (fingerprint, TouchID, FaceID etc.) support available on it otherwise Device PIN/Passcode be used to protect the generated keys. If this flag is set to TRUE and the device doesn't support biometric functionality, then an error should be generated by mobile app.
allowed_devices	MANDATORY	*Integer*	It defines the limit of devices to be registered. Default value is 0 for unlimited devices (max. 500 characters).
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

4.15 Generate QR Code

This API will be used by the business application to generate a QR Code using the Unity Service. The Unity Service will generate a QR Code image for a user and send in response. The business application can display this QR code where user can scan it on his/her mobile device for authentication.

https://<server>:8778/adss/service/unity/v1/authentication/qrcode/{userID}?{format=png}&{size=264}		
HTTP Verb	GET	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {user_access_token}	
Request Body		
Status Code	**Message**	**Response Body**

200	OK	{
		"format":"png",
		"size":"264",
		"qr_code":"<base64 encoded image>"
		}

400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 52 – Generate QR Code

Request Parameters

Parameters	Presence	Value	Description
{userID}	MANDATORY	*String*	User ID for whom the QR Code will be generated (max. 50 characters).
size	OPTIONAL	*Integer*	Size of the QR Code image in pixels. Since QR Code is in a square shape the parameter "size" will be used for both width and height of the image. Default size is 264 pixels
format	OPTIONAL	*String*	Format of the QR Code image e.g. png/jpeg/bmp etc. Default format will be "png" (max. 10 characters). The following formats are supported: • png • jpg • bmp • jpeg • wbmp • gif

Response Parameters

Parameters	Presence	Value	Description
format	MANDATORY	*String*	Format of the QR Code image e.g. png/bmp/jpg etc (max. 10 characters).
size	MANDATORY	*Integer*	Size of the QR Code image.
qr_code	MANDATORY	*String*	Base64 encoded image of the QR Code.

4.16 Verify QR Code

This API will be used to verify a QR Code by Unity Service if user set the authentication mechanism QR code in Unity profile. Mobile app can use the QR code reader to scan the QR code. If QR code is verified successfully, the Unity Service will return the access and refresh tokens in response.

https://<server>:8778/adss/service/unity/v1/authentication/qrcode		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {application_access_token}	

Request Body	{
	"user_id":"jhon123",
	"qr_code":"Information extracted from QR code"
	
	}	

Status Code	**Message**	**Response Body**

200	OK	{
		"access_token":"2YotnFZFEjr1zCsicMWpAA",
		"refresh_token":"TRVFHTHcedfJGJFLGKKJ",
		"expires_in":3600
		}

401	Unauthorized	If QR code is invalid or expired
400	Bad Request	
500	Internal Server Error	

429	Too Many Requests	{
		"error": "58129",
		"error_description": "Failed to process request - You have exhausted your API Request Quota"
		}

Table 53 – Verify QR Code

Request Parameters

Parameters	Presence	Value	Description
user_id	MANDATORY	*String*	User ID that identifies the user (max. 50 characters).
qr_code	MANDATORY	*String*	The QR code that needs to be verified.

Response Parameters

Parameters	Presence	Value	Description
access_token	MANDATORY	*String*	Access Token generated after verifying the QR code.

refresh_token	MANDATORY	*String*	Refresh token will be used to get the new access token without sending any credentials.
expires_in	MANDATORY	*String*	It's token expiry mentioned in the seconds.

4.17 Register Device for Push Notification

This API is used to register the mobile device for push notification by Unity Service. It takes the device token from the mobile application and stores in ADSS Unity Service to send the push notification while generating the authorization request.

https://<server>:8778/adss/service/unity/v1/authorization/push/notification		
HTTP Verb	POST	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {user_access_token}	

Request Body	{
	"device_token":"2YotnFZFEjr1zCsicMWpAA ",
	"os_type":"ANDROID/IOS"
	
	}	

Status Code	**Message**	**Response Body**
200	OK	
401	Unauthorized	Invalid or expired user access token
400	Bad Request	Device token is missing
500	Internal Server Error	
429	Too Many Requests	

Table 54 – Register Device for Push Notification

Request Parameters

Parameters	Presence	Value	Description
device_token	MANDATORY	*String*	It's the device token which needs to be sent on server to register the mobile device for push notification.
os_type	MANDATORY	*String*	OS type can be Android or iOS (max. 50 characters).

4.18 Delete Device for Push Notification

This API is used to delete the registered mobile device for push notification by UNITY Service.

https://<server>:8778/adss/service/unity/v1/authorization/push/notification/{device_token}		
HTTP Verb	DELETE	
Content-Type	application/json	
Accept	application/json	
Authorization	Bearer {device_token}	

Request Body	{
	"device_token":"2YotnFZFEjr1zCsicMWpAA "
	}	

Status Code	**Message**	**Response Body**
200	OK	
429	Too Many Requests	

Table 55 – Delete Device for Push Notification

Request Parameters

Parameters	Presence	Value	Description
device_token	MANDATORY	*String*	It's the device token which needs to be sent on server to register the mobile device for push notification.

Response Parameters

Parameters	Presence	Value	Description
error_code	CONDITIONAL	*String*	The error code.
error_description	CONDITIONAL	*String*	Error description message.

5 Signature Activation Data (SAD) – Body Structure

The body structure of SAD XML is explained in the table below:

Body Structure	<AuthorisationData> <OriginatorID>Virtual_CSP_Client</OriginatorID> <UserID>olcayatli@gmail.com</UserID> <CertificateID>416edc72-6c63-45aa-bb34a373102234df</CertificateID> <TransactionID>980551837300673581</TransactionID> <Salt>924552495291565632</Salt> <MetaData> <DisplayText>Data to be displayed</DisplayText> <DeviceID>ad1e60a6-5e23-4b52-a127-27f41c224c05</DeviceID> </MetaData> <NumSignatures>1</NumSignatures> <Documents> <Document id="b81e040a-a4d8-4134-92ff-2d4bf5e9116d"> <Name>b81e040a-a4d8-4134-92ff-2d4bf5e9116d</Name> <DigestValue>ypkP9L2tZO2JdfNr4X4X5SRur529uJqykdc5q5HDSiLNiYcLrysO0S/H31yb8QZS
SOBYsFlVSj9/SKUqrhsUC5oEc/gr</DigestValue> </Document> </Documents> <ValidityPeriod> <ValidFrom>2019-12-07T18:25:37</ValidFrom> <ValidTo>2019-12-07T18:42:17</ValidTo> </ValidityPeriod> <Signature> <DigestMethod>SHA256</DigestMethod> <SignatureValue>MEUCID/kiJWAIqzwOp/hi+FUbJwsjdcsEoBNwlIF8sXA8XbDAiEAkGgQomyoJ2iR0ra9KGBFW/zXi6tbsn5M49YiaPNc+L8=</SignatureValue> </AuthorisationData>

The body structure of SAD JSON is explained in the table below:

Body Structure	**Header:**
	{
	"typ": "JWT",
	"alg": "ES256K"
	}
	Payload:
	{
	"transaction_id": "890011353477997427",
	"originator_id": "samples_test_client",
	"user_id": "am2",
	"certificate_id": "am2-key",
	"salt": "735775875109545617",
	"device_id": "",
	"number_signatures": "2",
	"documents": [
	{
	"document_name": "",
	"digest_value":
	"2467c291078b4d1dcf789ebe24b10ba50add1195048de5b060fe29a17398eda7"
	}
	],
	"validity_period": {
	"valid_from": "2023-08-07T17:52:53",
	"valid_to": "2023-08-08T01:51:43"
	}
	}
	JSON Web Signature:
	eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NksifQ.eyJ0cmFuc2FjdGlvbl9pZCI6Ijg5MDAxMTM1MzQ3Nzk5NzQyNyIsIm9yaWdpbmF0b3JfaWQiOiJzYW1wbGVzX3Rlc3RfY2xpZW50IiwidXNlcl9pZCI6ImFtMiIsImNlcnRpZmljYXRlX2lkIjoiYW0yLWtleSIsInNhbHQiOiI3MzU3NzU4NzUxMDk1NDU2MTciLCJkZXZpY2VfaWQiOiIiLCJudW1iZXJfc2lnbmF0dXJlcyI6IjIiLCJkb2N1bWVudHMiOlt7ImRvY3VtZW50X25hbWUiOiIiLCJkaWdlc3RfdmFsdWUiOiIyNDY3YzI5MTA3OGI0ZDFkY2Y3ODllYmUyNGIxMGJhNTBhZGQxMTk1MDQ4ZGU1YjA2MGZlMjlhMTczOThlZGE3In1dLCJ2YWxpZGl0eV9wZXJpb2QiOnsidmFsaWRfZnJvbSI6IjIwMjMtMDgtMDdUMTc6NTI6NTMiLCJ2YWxpZF90byI6IjIwMjMtMDgtMDhUMDE6NTE6NDMifX0.Yf-b-GbYnz1689b7iqHwkRNeOvCpg_aW7_AGJ_I5TySkoOnu2Ba64_qBhs6BAN-kfQ7hJzfIdZal4j6J3oA5Qw

6 Updates

No updates were incorporated from the previous to the current version of ADSS Server.

7 Error Code List

Below table contains the error codes for UNITY business and mobile interfaces.

Errors	
error	**error_description**
65001	An internal server error occurred while processing the request - see the Unity service debug logs for details
65002	Failed to process request - Unity service is stopped
65003	Failed to process request - Unity service is not enabled in license
65004	Failed to process request - Unity service license has expired
65005	Failed to process request - Unity service is not enabled in system
65006	Failed to process request - Unity service is not allowed
65007	Failed to process request - Client ID does not exist
65008	Failed to process request - User ID does not exist
65009	Failed to process request - User ID already exists
65010	Failed to process request - Key alias does not exist
65011	Failed to process request - Transaction ID does not exist
65012	Failed to process request - Client ID not found in the request
65013	Failed to process request - User ID not found in the request
65014	Failed to process request - Key alias not found in the request
65015	Failed to process request - Subject DN not found in the request
65016	Failed to process request - User password not found in the request
65017	Failed to process request - Key length not found in the request
65018	Failed to process request - Key algorithm not found in the request
65019	Failed to process request - User name not found in the request
65020	Failed to process request - User password not found in the request
65021	Failed to process request - User mobile number not found in the request
65022	Failed to process request - Key alias exceeds the allowed limit
65023	Failed to process request - User ID exceeds the allowed limit
65024	Failed to process request - User name exceeds the allowed limit
65025	Failed to process request - User password exceeds the allowed limit
65026	Failed to process request - Invalid user mobile number

65027	Failed to process request - Invalid user email
65028	Failed to process request - Invalid user status
65029	Failed to process request - Unity profile does not exist or marked inactive
65030	Failed to process request - User certificate not found in the request
65031	Failed to process request - Profile ID not found in the request
65032	Failed to process request - Invalid client ID
65033	Failed to process request - User's new password not found in the request
65034	Failed to process request - SMS OTP not found in the request
65035	Failed to process request - Email OTP not found in the request
65036	Invalid string parameter - refresh_token
65037	Failed to process request - Invalid refresh token
65038	Failed to process request – Invalid/expired access token
65039	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.
65040	Failed to process request - Basic authentication is not enabled in Unity profile
65041	Failed to process request - SAML authentication is not enabled in Unity profile
65042	Failed to process request - Missing (or invalid type) string parameter token
65043	Failed to process request - Invalid string parameter token_type_hint
65044	Failed to process request - Invalid string parameter token
65045	Failed to process request - Client ID is not configured for certification service settings in Unity service manager
65046	Failed to process request - Certification profile is not configured for certification service settings in Unity service manager
65047	Failed to process request - Certification service address is not configured for certification service settings in Unity service manager
65048	Failed to process request - Unable to get device certificate from certification service
65049	Failed to generate access token - HMAC Key not configured in Unity service manager
65050	Failed to process request - User Email not found in the request

65051	Failed to process request - Client ID is not configured for default settings in Unity service manager
65052	Failed to process request - Device ID not found in the request
65053	Failed to process request - Push notification token not found in the request
65054	Failed to process request - OS type not found in the request
65055	Missing (or invalid type) string parameter credentialID
65056	Missing (or invalid type) integer parameter numSignatures
65057	Invalid parameter numSignatures
65058	Invalid request parameter - numSignatures doesn't match with no of hashes in hash array
65059	Invalid request parameter - no of documents in clientData doesn't match with no of hashes in hash array
65060	Missing parameter hash
65061	Failed to authorise user credentials - Request timeout for mobile authorization
65062	Failed to authorise user credentials - User cancelled mobile authorization
65063	Invalid parameter credentialID
65064	Missing (or invalid type) string parameter SAD
65065	Invalid parameter SAD
65066	Empty hash array
65067	Invalid Base64 hash string parameter
65068	Missing (or invalid type) string parameter signAlgo
65069	Invalid parameter signAlgo
65070	Missing (or invalid type) string parameter hashAlgo
65071	Invalid parameter hashAlgo
65072	Failed to validate SAML assertion - Invalid base64 data
65073	Failed to validate SAML assertion - Not comply with SAML 2.0 schema
65074	Failed to validate SAML assertion - Unable to parse SAML assertion
65075	Failed to validate SAML assertion - Validity period expired or not yet valid
65076	Failed to validate SAML assertion - Server certificate does not match with the certificate configured in Unity Profile
65077	Failed to validate SAML assertion - Multiple or no attributeValue found

65078	Failed to validate SAML assertion - Invalid Signature
65079	Failed to process request - User status is blocked or inactive
65080	Failed to process request - Certificate chain not found in the request
65081	Failed to process request - Invalid certificate chain
65082	Failed to process request - Client secret not found in the request
65083	Failed to authorise user credentials - An internal server error occurred during signature computation
65084	Failed to process request - Device CSR not found in the request
65085	Failed to process request - Invalid device CSR
65086	Failed to process request - Device information not found in the request
65087	Failed to process request - Device ID not found in the request
65088	Failed to process request - Device name not found in the request
65089	Failed to process request - SAD not found in the request
65090	Failed to process request - Request ID not found in the request
65091	Failed to process request - Invalid request
65092	Failed to process request - Either request ID is invalid or the transaction is expired
65093	An internal server error occurred - please contact your service provider
65094	Failed to process request - User mobile exceeds the allowed limit
65095	Failed to process request - User email exceeds the allowed limit
65096	Failed to process request - Configurations for SMS/Email OTP(s) not available
65097	Failed to process request - No OTP(s) found in request
65098	Failed to process request - QR Code authentication is not allowed for this Unity profile
65099	Failed to process request - QR Code is invalid or expired
65100	Failed to process request - Missing parameter Grant type
65101	Failed to process request - Invalid parameter Grant type
65102	Failed to process request - Missing parameter authorization code
65103	Failed to process request - Invalid parameter authorization code
65104	Failed to process request - Missing parameter Redirect URI

65105	Failed to process request - Missing Redirect URI
65106	Failed to process request - Authorization code is invalid or expired
65107	Failed to process request - Scope is missing or invalid
65108	Failed to process request - Response type is missing or invalid
65109	Missing (or invalid type) integer parameter certificates
65110	Information message is not present for the credential authorization page
65111	Failed to validate signature activation data - XML schema validation failed
65112	Failed to validate signature activation data - Failed to parse the XML
65113	Failed to process request - Missing parameter 'PIN'
65114	Failed to process request - Missing or invalid parameter 'OTP'
65115	Failed to process request - Missing or invalid parameters 'PIN' and 'OTP'
65116	Failed to process request - Invalid PIN
65117	Failed to process request - OTP is invalid or expired
65118	Failed to Sign SAD - Authorisation Certificate is not configured in Unity profile
65119	Failed to process request - Explicit credentials authorisation with OTP must be selected in profile
65120	Failed to process request - Extend transaction is not allowed. Profile is set to require user authorisation for every transaction
65121	Failed to process request - The limit of devices you can register has been reached
65122	Failed to process request - External IdP token id expired
65123	Failed to process request - Configured connector in Unity profile is inactive
65124	Failed to process request - Authorisation Certificate is not yet valid
65125	Failed to process request - Authorisation Certificate is expired
65126	Failed to process request - number of hashes provided is greater than the numSignatures value
65127	Failed to process request - Unity Credentials authorisation profile settings are not compatible with SAM profile SCAL level.
65128	Failed to process request - Start Index is invalid in the request
65129	Failed to process request - Fetch Size is invalid in the request
65130	Failed to process request - Order is invalid in the request

65131	Failed to process request - You have exhausted your API Request Quota
65132	Failed to process request - Description exceeds the allowed limit
65133	Failed to process request - OAuth2 is not allowed in Service Authorisation Settings for this Unity profile
65134	Failed to process request - Authorisation request not found against the transaction ID
65135	Failed to process request - Authorisation request is expired
65136	Failed to process request - BrainPool R and T curves are not supported for credentials authorisation in case of JSON based SAD
65137	Failed to process request - Key length of 256 or higher for ECDSA key is only supported for credentials authorisation in case of JSON based SAD
65138	Failed to process request - User Authentication is required for this Unity profile
65139	Failed to process request - Missing parameter AuthData
65140	Failed to process request - Invalid handle request parameter
65141	Failed to process request - Client authorisation is required
65142	Failed to process request - Missing parameter AuthObjectID
65143	Failed to process request - Invalid request parameter authObjectID
65144	Failed to process request - Invalid request parameter scope
65145	Failed to process request - Invalid request parameter signatureQualifier
65146	Failed to process request - Invalid request parameter with authorization_details parameter
65147	Failed to process request - Invalid request parameter authorization_details
65148	Failed to process request - Missing parameter code_challenge
65149	Failed to process request - request_uri should not be present
65150	Failed to process request - Missing or invalid Authorization Details parameter
65151	Invalid Request - Missing or Empty documentDigests and documents objects
65152	Invalid Request - Both documentDigests and documents parameters passed
65153	Invalid Request - Invalid object parameter documentDigests
65154	Invalid Request - Invalid Base64 hashes string parameter
65155	Invalid Request - Invalid Base64 documents string parameter

65156	Invalid Request - Invalid Pdf document
65157	Failed to sign Pdf document - Signed document can not be encrypted
65158	Failed to sign Pdf document - PDF is already certify signed
65159	Failed to sign Pdf document - A certify signed PDF with no changes allowed cannot be signed
65160	Failed to sign Pdf document - Signing field not found in the PDF document
65161	Failed to sign Pdf document - Signing field is not configured in the profile
65162	Failed to sign Pdf document - Input document is encrypted
65163	Failed to sign Pdf document - A signed PDF cannot subsequently be certified signed
65164	Failed to sign Pdf document - See the Unity service debug logs for details
65165	Failed to sign Pdf document - Problem in creating/signing empty signature field
65166	Failed to sign Pdf document - Failed to create visible signature
65167	Failed to sign Pdf document - Failed to create invisible signature
65168	Failed to sign Pdf document - Page does not exist
65169	Failed to create CAdES Signatures - See the Unity service debug logs for details
65170	Invalid Request - Invalid signature format. Supported format are 'P', 'C' and 'X' for PAdES, CAdES and XAdES
65171	Failed to enhance CAdES Signatures - See the Unity service debug logs for details
65172	Invalid Request - String parameter hashAlgorithmOID contradicts with signAlgo parameter
65173	Invalid Request - Missing or invalid type string parameter signAlgoParams
65174	Invalid Request - Missing string parameter signature_format
65175	Invalid Request - Conformance level is invalid. supported levels are 'AdES-B-B', 'AdES-B-T', 'AdES-B-LT', 'AdES-B-LTA'
65176	Invalid Request - Invalid parameter signed envelope property
65177	Invalid Request - Invalid parameter signed envelope property for requested signature format
65178	Invalid parameter signed properties - list of invalid attributes
65179	Failed to sign - Signature Format is not allowed in profile

65180	Failed to sign - Signed Envelope property is not allowed in profile
65181	Failed to sign - Conformance Level is not allowed in profile
65182	Failed to sign - Document size in request exceeds to the allowed limit in profile
65183	Failed to sign - Document Digest size in request exceeds to the allowed limit in profile
65184	Invalid Request - Invalid value for sig_area parameter. Possible values are 'TOP_LEFT', 'TOP_RIGHT', 'CENTER', 'BOTTOM_LEFT', 'BOTTOM_RIGHT'
65185	Invalid Request - Invalid value for sig_page parameter. Must be a non zero integer
65186	Invalid digest value length
65187	Empty hash parameter
65188	Invalid parameter nonce
65189	Failed to enhance PAdES signatures - See the Unity service debug logs for details
65190	Failed to create CAdES signatures - Signed Property commitment-type-indication is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.3 of ETSI EN 319 122-1
65191	Failed to create CAdES signatures - Signed Property content-hints is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.4.1 of ETSI EN 319 122-1
65192	Failed to create CAdES signatures - Signed Property mime-type is not valid Base64 encoded UTF8String as defined in clause 5.2.4.2 of ETSI EN 319 122-1
65193	Failed to create CAdES signatures - Signed Property signer-location is not valid Base64 encoded UTF8String as defined in clause 5.2.5 of ETSI EN 319 122-1
65194	Failed to create CAdES signatures - Signed Property content-timestamp is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.8 of ETSI EN 319 122-1
65195	Failed to create CAdES signatures - Signed Property signer-attribute-v2 is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.6.1 of ETSI EN 319 122-1
65196	Failed to create CAdES signatures - Signed Property content-reference is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.11 of ETSI EN 319 122-1
65197	Failed to create CAdES signatures - Signed Property signature-policy-identifier is not valid Base64 encoded ASN.1 Sequence as defined in clause 5.2.9.1 of ETSI EN 319 122-1

65198	Failed to create CAdES signatures - Signed Property content-identifier is not valid Base64 encoded ASN.1 Octet String as defined in clause 5.2.12 of ETSI EN 319 122-1
65199	Failed to process the request – Credentials ID and Signature Qualified can not be used together
65200	Invalid Request - Detached Signed Envelop property is not supported for XAdES
65201	Failed to process request - Full document signing is disabled in profile
65202	Failed to create XAdES signature - Signing Element not found in xml
65203	Failed to compute data to be signed for XAdES signature
65204	User ID mismatch: Provided user ID does not match the user ID in the token.
65205	Invalid Request - Malformed authorization header. It does not match the pattern 'Bearer [sessionKey]'
65206	Failed to create XAdES Signatures - See the Unity service debug logs for details
65207	Failed to enhance XAdES Signatures - See the Unity service debug logs for details
65208	Failed to create XAdES Signatures - embedded data in time stamp is invalid
65209	Failed to create XAdES Signatures - embedded data in signature is invalid
65210	Failed to create XAdES signatures - Signed Property SignerRoleV2 is not a valid Base64 encoded xml element as defined in clause 5.2.6 of ETSI EN 319 132-1
65211	Failed to create XAdES signatures - Signed Property CommitmentTypeIndication is not a valid Base64 encoded xml element as defined in clause 5.2.3 of ETSI EN 319 132-1
65212	Failed to create XAdES signatures - Signed Property SignatureProductionPlaceV2 is not a valid Base64 encoded xml element as defined in clause 5.2.5 of ETSI EN 319 132-1
65213	Failed to create XAdES signatures - Signed Property AllDataObjectsTimestamp is not a valid Base64 encoded xml element as defined in clause 5.2.8.1 of ETSI EN 319 132-1
65214	Failed to create XAdES signatures - Signed Property IndividualDataObjectsTimestamp is not a valid Base64 encoded xml element as defined in clause 5.2.8.2 of ETSI EN 319 132-1
65215	Failed to create XAdES signatures - Signed Property SignaturePolicyIdentifier is not a valid Base64 encoded xml element as defined in clause 5.2.9 of ETSI EN 319 132-1
65216	Failed to create XAdES signatures - Document Digest is not supported for XAdES Signature

65217	Failed to create PAdES signatures - 'No Changes Allowed' Certify option is not allowed for PAdES-B-LT or PAdES-B-LTA
65218	Failed to create Pdf Multi-Signature - Extend Transaction failed
65219	Invalid EPES signature configurations. Failed to find signature policy file
65220	Failed to create PAdES signatures. Certified signature can only be created with one precise signing locations using Pdf drawer settings
Common Errors	
86000	Failed to authenticate client - TLS client authentication certificate has expired
86001	Failed to authenticate client - TLS certificate CN does not match with Client ID
86002	Failed to authenticate client - TLS client certificate is revoked
86003	Failed to authenticate client - revocation status for TLS client certificate is unknown
86004	Failed to authenticate client - Client ID does not match with the client identified by TLS client certificate
86005	Failed to authenticate client - TLS client certificate does not match with the configured client certificate
86006	Failed to authenticate client - request signing certificate has expired
86007	Failed to authenticate client - request signing certificate is revoked
86008	Failed to authenticate client - revocation status for request signing certificate is unknown
86009	Failed to authenticate client - request signing certificate does not match with the configured client certificate
86010	Failed to authenticate client - Client ID does not match with the client identified by the request signing certificate
86011	Failed to authenticate client - Client ID does not exist
86012	An error occurred while communicating with database - see the service debug logs for details
86013	An error occurred when checking the certificate revocation status - see the service debug logs for details
86014	An internal error occurred while authenticating the client - see the service debug logs for details
86015	Failed to authenticate client - Client ID is not found in the request
86016	Failed to process request - Request signing certificate is not trusted

86017	Failed to authenticate client - client is marked inactive
86018	Failed to authorise client - service is not allowed to this client
86019	Failed to authorise client - service profile does not exist
86020	Failed to authorise client - service profile is inactive
86021	Failed to authorise client - profile is not allowed to this client
86022	Failed to authorise client - default profile not configured and neither found in request
86023	Failed to authorise client - default profile is inactive
86024	Failed to authorise client - client secret is invalid
internal_error	An internal server error occurred while processing the request
invalid_csr	CSR is invalid
invalid_otp	OTP is either invalid or expired
missing_csr	CSR is missing in the request
missing_device_id	Device ID is missing in the request
missing_device_info	Device information is missing in the request
missing_device_name	Device name is missing in the request
missing_request_id	Request ID is missing in the request
refresh_token_revoked	Refresh token is either invalid or expired

Table 56 – Error Codes

*** End of Document ***