CTX-User-Access-Management User Guide

Contents

Versions

Document Revisions

The following revisions have been made to this document

Date	Revision	Notes
04/01/2019	1.0	First release

Module Versions

The following revisions have been made to this document

Date	Revision	Notes
04/01/2019	1.0	Creation of: • UAM-Authenticate-User • UAM-End-User-Session • UAM-SessionManagement • UAM-Create-User-Session • UAM-Check-Authorisation-Token

Preface

About this Manual

This document is a user guide for the CTX-User-Access-Management module.

Audience

The audience for this document is those wanting to understand how to use CTX-User-Access-Management module.

Related Material

Document
CTX-User-Access-Management – Deployment Plan
CTX-User-Access-Management.studiopkg
CTX-Vista-User-Management – Deployment Plan

Abbreviations used in this Document

OCI	Orchestration Communication Interface
UAM	User Access Management
AD	Active Directory

Requirements

The CTX-User-Access-Management module requires the following:

- Cortex Database OCI
- If to be used with Active Directory integration:
  - Cortex Active Directory OCI
- If to be used with Cortex Vista security integration:
  - Cortex PowerShell OCI
  - CTX-Vista-User-Management module

Instructions for how to install these are included in the deployment plan.

Integration

Integration with Third-Party Systems

User Access Management Database

For the flows and subtasks to work in the CTX-User-Access-Management module, the Cortex User Access Management database and schema needs to exist on the server containing the Cortex databases. Instructions how to set this up are provided in the 'CTX-User-Access-Management – Deployment Plan'.

The tables involved in the Cortex User Access Management schema are:

- User – Table containing the details of the user
- Session – Stores all the user session information:
    - Authentication token
    - Start Time
    - End Time (Only when Inactive)
    - Status (Active or Inactive)
- Activity – Stores all the session activity. Every time the user token is checked, via the UAM-Check-Authorisation-Token subtask, the below information is logged:
    - Time
    - Flow execution unique identifier

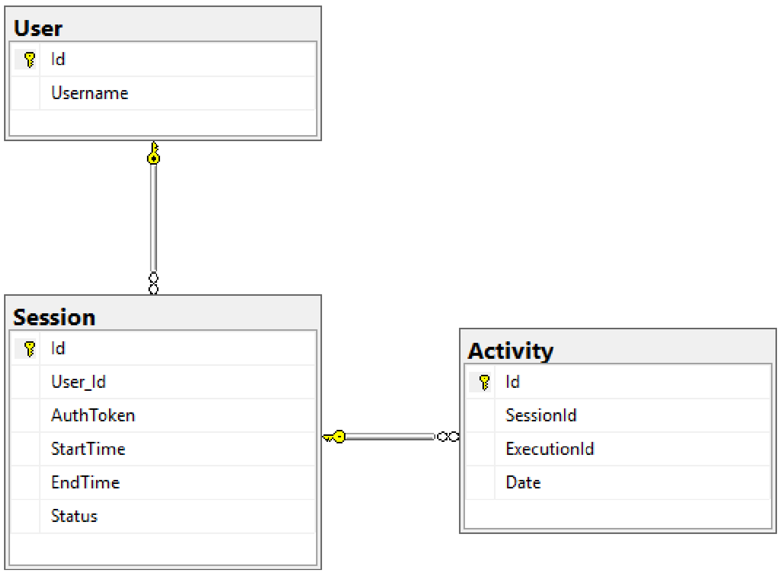


Figure 1 - Cortex User Access Management database schema

Integration with Existing Infrastructure

CTX-Vista-User-Management

For the CTX-User-Access-Management module to work with Cortex Vista Security, the CTX-Vista-User-Management module needs to be installed on the server. Instructions how to set this up are provided in the 'CTX-Vista-User-Management – Deployment Plan'.

1 User Access Management Overview

The User Access Management module should be used as the single authority to manage authentication and authorisation within a system:

- **Authentication**

  User provides valid credentials to gain initial access to a system. When the user is authenticated, a session is created, and this session should then be referred, by using the authentication token, during any interactions between the user and the system.

  Authentication can be done against an Active Directory domain or Cortex Vista Security. There is no restriction on using one or the other, or both.

- **Authorisation**

  Determines whether a user has access to a specific resource of the system.

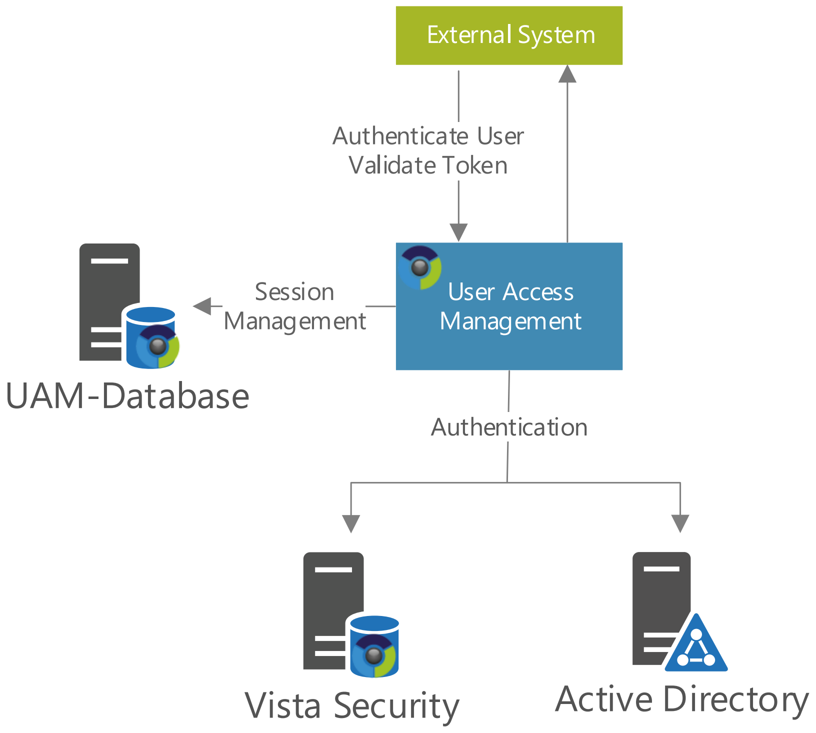


Figure 2 – UAM Architecture

1.1.1 Using the module

The User Access Management module can be used by external systems to manage authentication and authorisation, without the need to develop specific interfaces to databases, active directories or authorisation systems.

As specified in the diagram below, the external system will authenticate against the Cortex UAM, using the UAM-Authenticate-User flow, and then use the temporary authorisation token, until it expires, on all following communications with the Cortex Solution flows. This means the external system most sensitive data (username and password) are only shared over the network at the start of the communication.

Once the Cortex Solution flows receive incoming requests with the authorisation token from the external system they must use the UAM-Validate-Authorisation-Token subtask to validate the token with the Cortex UAM.

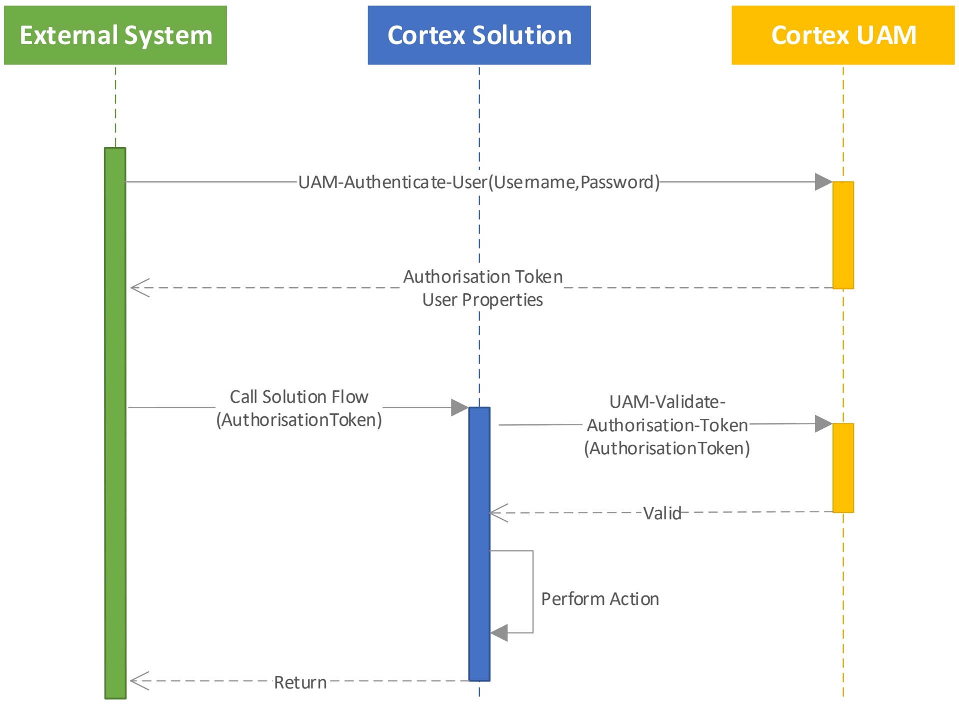


Figure 3 - UAM as authentication and authorisation entity within a Cortex Solution

2 User Access Management Flows

2.1 UAM-Authenticate-User

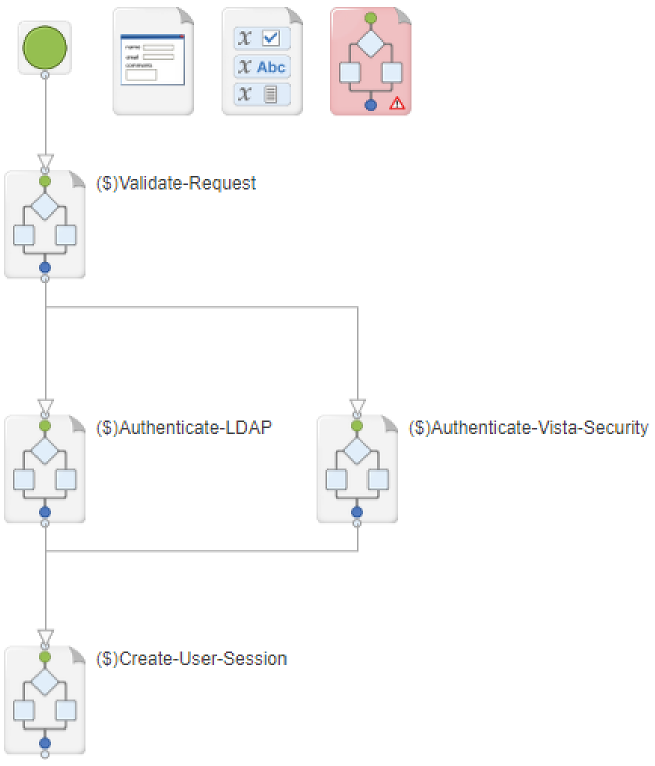
2.1.1 Overview

Authenticates a user in Active Directory or Vista security and generates an authentication token. The flow extracts the domain from the username variable. If the domain is Vista, the authentication is done against the Cortex Internal Zebedee database. If any other domain, the authentication is done against active directory.

Exceptions will be raised if:

- Username is not supplied
- Username format is not domain\username
- Password is not supplied
- Credentials are incorrect
- The Active Directory domain server (if AD access) is not accessible
- The Cortex Internal Zebedee database (if Vista security) is not accessible
- The User Access Management database is not accessible

2.1.2 States



- Validate-Request

Validates that the Username and Password inputs are passed into the flow. Also validates that the Username format is domain\username. When the domain is Vista, the flow branches to the Authenticate-Vista-Security state; For any other domain, the flow branches to the Authenticate-LDAP.

- Authenticate-LDAP

  Connects to the Active Directory (AD) domain extracted from the username and validates the account credentials. If successful, retrieves the name, email and user groups from AD. If not successful, raises an exception.

- Authenticate-Vista-Security

  Connects to the Cortex Internal Zebedee database and validates account credentials. If successful, retrieves the name, email and user ACEs from the database. If not successful, raises an exception.

- Create-User-Session

  Calls the UAM-Create-User-Session subtask, specified in 3.1, to create the session and generate an authentication token.

2.1.3 Inputs

Input Variables	Type	Description
i_Username	Text	The user to be authenticated. Format is domain\username. REQUIRED Example: cortex\user
i_Password	Text	The password for the user to be authenticated. Ideally the password should be Cortex encrypted, although it can be passed non-encrypted. REQUIRED Examples: • #_046058104069144!124137050078078239025218131019183~045080153170026!191241213208091022113131213157025# • P4ssw0rd
i_SQL-Server	Text	The server where the Cortex Access User Management database is hosted. Default value is set to 'localhost' Example: localhost
i_DB-Name	Text	The name of the Cortex Access User Management database. Default value is set to 'Cortex-UserAccessManagement' Example: Cortex-UserAccessManagement

2.1.4 Outputs

Output Variables	Type	Description
o_Authentication-Toke	Text	The generated authentication token. Example: 28104AC4-0C44-4F45-9814-E0945CF4125E

o_User-Properties	Structure	Contains the following user properties: Username, Name, Email, Security Groups
		
		Example:
		
		{
		  "NAME": "System Administrator",
		  "MAIL": "administrator.user@pivetal.com",
		  "USERNAME": "administrator",
		  "ROLES": [
		    "Administrators",
		    "Users"
		  ]
		}

o_Status	Structure	Contains the success or exception message.
		
		Example:
		
		If ok:
		
		{
		  "STATUS": "OK"
		}
		If exception:
		
		{
		  "STATUS": "Exception",
		  "EXCEPTION_CODE": "1",
		  "SHORT_DESCRIPTION": "The username or password is incorrect.",
		  "FULL_DESCRIPTION": "The username or password is incorrect.",
		  "RAW_EXCEPTION": "19 Oct 2018 5:06:35 p.m.: Block Type: SUBTASK, ID=0,\r\nBlock Description: ,\r\nBlock UUID: 772edb5d697c4283a10172ff3de1a860.\r\n\r\nError Message: The username or password is incorrect.\r\n-----------------------\r\n19 Oct 2018 5:06:35 p.m.: Block Type: SIGNAL-ERROR, ID=0,\r\nBlock Description: Raise exception,\r\nBlock UUID: 4cf58981054d44e7b45078304260d6b9.\r\n\r\nError

		Message: The username or password is incorrect.\r\n-----------------------\r\n", "TIMESTAMP": "19-Oct-2018 17:06:36", "FLOW_NAME": "UAM-AUTHENTICATE-USER", "EXECUTION_UUID": "f2c3896ad3b811e8966900505691778f" }

2.2 UAM-End-User-Session

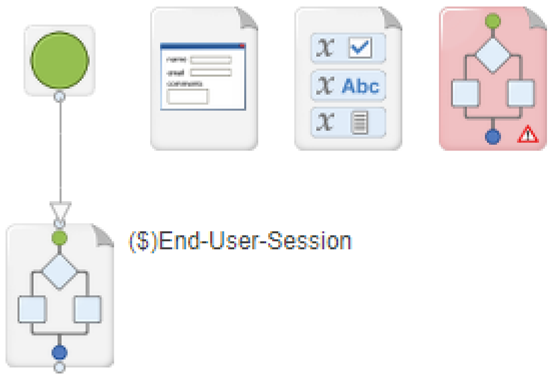
2.2.1 Overview

The UAM-End-User-Session ends a user session in the User Access Management database by making the authentication inactive.

Exceptions will be raised if

- The User Access Management database is not accessible

2.2.2 States



- End-User-Session

  Connects to the User Access Management database and makes the authentication token inactive.

2.2.3 Inputs

Input Variables	Type	Description
i_SQL-Server	Text	The server where the Cortex Access User Management database is hosted. Default value is set to 'localhost' Example: localhost
i_DB-Name	Text	The name of the Cortex Access User Management database. Default value is set to 'Cortex-UserAccessManagement' Example: Cortex-UserAccessManagement
i_Authentication-Token	Text	The session authentication token. REQUIRED Example: 28104AC4-0C44-4F45-9814-E0945CF4125E

2.2.4 Outputs

Output Variables	Type	Description
o_Status	Structure	Contains the success or exception message. Example: See example in section 2.1.4.

2.3 UAM-Session-Management

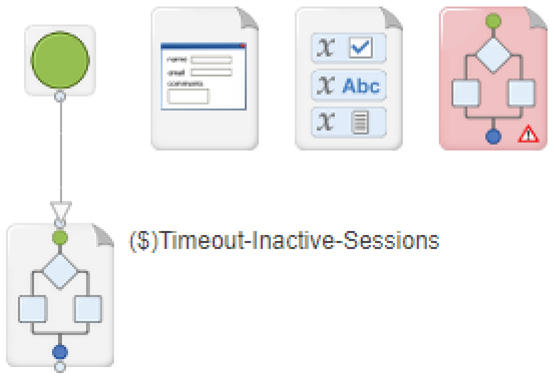
2.3.1 Overview

The UAM-Session-Management is used to timeout sessions that have been inactive for a period of time.

Exceptions will be raised if

- The User Access Management database is not accessible

2.3.2 States



- End-User-Session

  Connects to the User Access Management database and makes sessions inactive if these have not registered any activity for a period of time, defined by the input variable i_Time-Out-Threshold.

2.3.3 Inputs

Input Variables	Type	Description
i_SQL-Server	Text	The server where the Cortex Access User Management database is hosted. Default value is set to 'localhost' Example: localhost
i_DB-Name	Text	The name of the Cortex Access User Management database. Default value is set to 'Cortex-UserAccessManagement' Example: Cortex-UserAccessManagement
i_Time-Out-Threshold	Text	Timeout threshold in minutes. Any session that has not registered activity within the defined threshold time will be made inactive. Default value is set to 30 Example: 30

2.3.4 Outputs

Output Variables	Type	Description
o_Status	Structure	Contains the success or exception message. Example: See example in section 2.1.4.

3 User Access Management Subtasks

3.1 UAM-Create-User-Session

3.1.1 Overview

The UAM-Create-User-Session creates a new session in the User Access Management database and returns the authentication token generated.

Exceptions will be raised if

- The User Access Management database is not accessible

3.1.2 Inputs

Input Variables	Type	Description
CUS_i_SQL-Server	Text	The server where the Cortex Access User Management database is hosted. Default value is set to 'localhost' Example: localhost
CUS_i_DB-Name	Text	The name of the Cortex Access User Management database. Default value is set to 'Cortex-UserAccessManagement' Example: Cortex-UserAccessManagement
CUS_i_Username	Text	The user for which the authentication token will be generated REQUIRED Example: cortex.user

3.1.3 Outputs

Output Variables	Type	Description
CUS_o_Authentication-Token	Text	The generated authentication token. Example: 28104AC4-0C44-4F45-9814-E0945CF4125E

3.2 UAM-Check-Authorisation-Token

3.2.1 Overview

The UAM-Check-Authorisation-Token validates if an authentication token generated by the Access User Management is active.

Exceptions will be raised if

- The User Access Management database is not accessible

3.2.2 Inputs

Input Variables	Type	Description
CAT_i_SQL-Server	Text	The server where the Cortex Access User Management database is hosted. Default value is set to 'localhost' Example: localhost
CAT_i_DB-Name	Text	The name of the Cortex Access User Management database. Default value is set to 'Cortex-UserAccessManagement' Example: Cortex-UserAccessManagement
CAT_i_Authentication-Token	Text	The authentication token to be validated. REQUIRED Example: 6FE3A4FD-1010-4811-9C3B-74E537B803E3

3.2.3 Outputs

Output Variables	Type	Description
CAT_o_Valid	Boolean	Specifies if the token is valid. Example: True