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
28/08/2019	1.1	Updated document to reflect the fix on flow UAM-Authenticate-User
28/03/2022	1.2	Added additional how to use information

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
28/08/2019	1.1	Updated: • UAM-Authenticate-User ○ A bug has been fixed that caused the User Active Directory groups to be all the Domain groups instead of only the groups the User belonged to.

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
CTX-Vista-User-Management – User Guide
CTX-Vista-User-Management.studiopkg

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 the authentication system to be used is Active Directory, then the below OCI is required:
    - Cortex Active Directory OCI
- If the authentication system to be used is Cortex Vista Security, then the below OCI and Cortex GitHub module is required:
    - Cortex PowerShell OCI
    - CTX-Vista-User-Management module

Instructions for how to install these are included in the 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 solution:

- **Authentication**

  User provides valid credentials to gain initial access to a system. When the user is authenticated, a session is created. This is the session that should be referred to, when using the authentication token for 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.

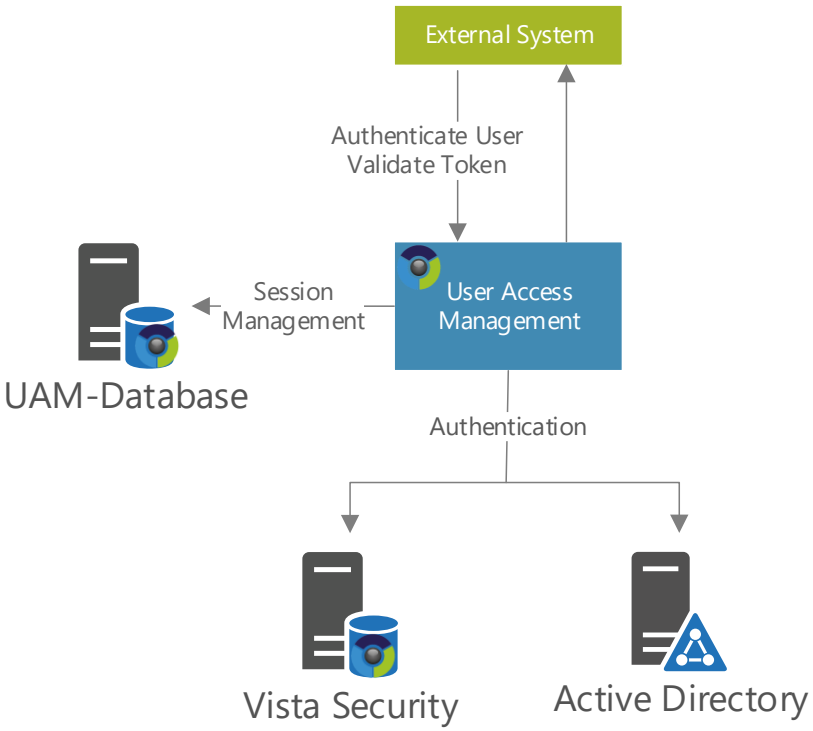


Figure 1 – UAM Architecture

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.

The next sections will exemplify some of the potential uses for this module.

1.1.1 External User Interface

A common use case for the Cortex User Access Management module is when using external user interfaces to communicate with Cortex.

As specified in the diagram below (Figure 2), the external user interface will authenticate against the Cortex UAM database, using the UAM-Authenticate-User flow. The flow will generate an authorisation token and look up the user properties, including the user roles.

The external user interface can use the looked-up roles to unlock features and allow specific actions depending on these. These features would be part of the user interface design and development.

After logging in, all communications between the user interface and the Cortex solution are required to use the temporary authorisation token. This means the user's 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 from the user interface with the authorisation token, it must use the UAM-Validate-Authorisation-Token subtask to validate the token with the Cortex UAM. Only valid authorisation tokens should be allowed to continue the flow execution.

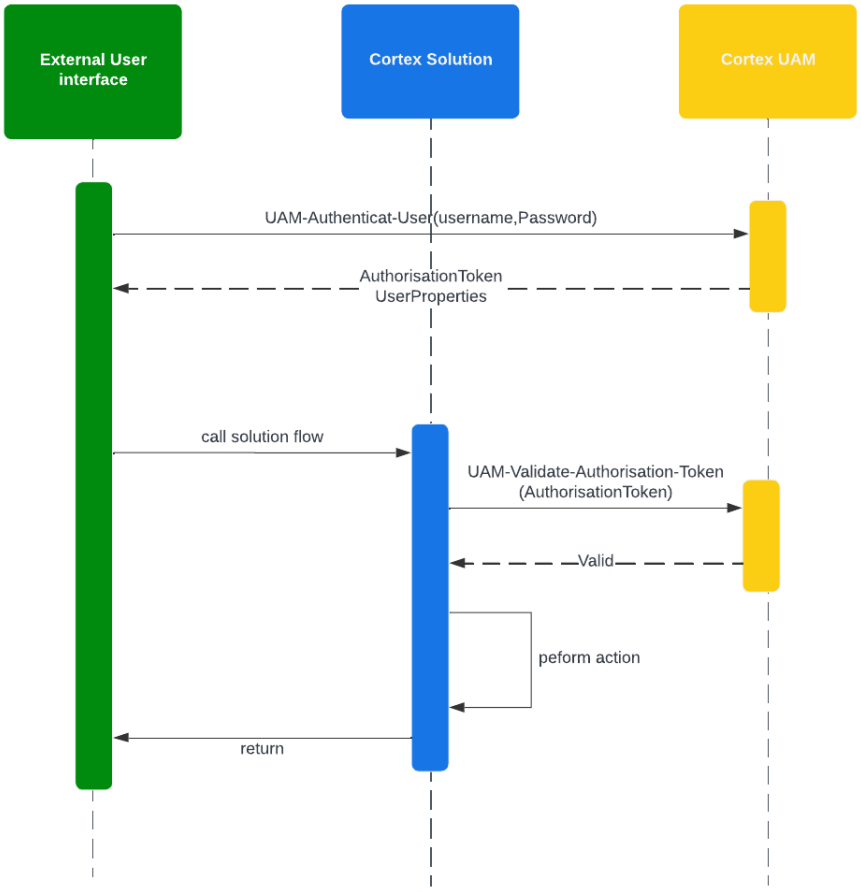


Figure 2 - UAM as authentication and authorisation entity within a Cortex Solution

1.1.2 System to system comunication authorisation

Another use case for this module is to authorise the communication between two different systems (Figure 3). Like the use case above (Figure 2), in the below use case (Figure 3) System 1 would generate an authorisation token from the Cortex UAM and then use it in the communications with System 2.

System 2 would validate this authorisation token against the Cortex UAM and only perform the action if a valid token was used. For this use it would be required to build an authorisation flow as explained in the next section (1.2 Implementation Recommendations)

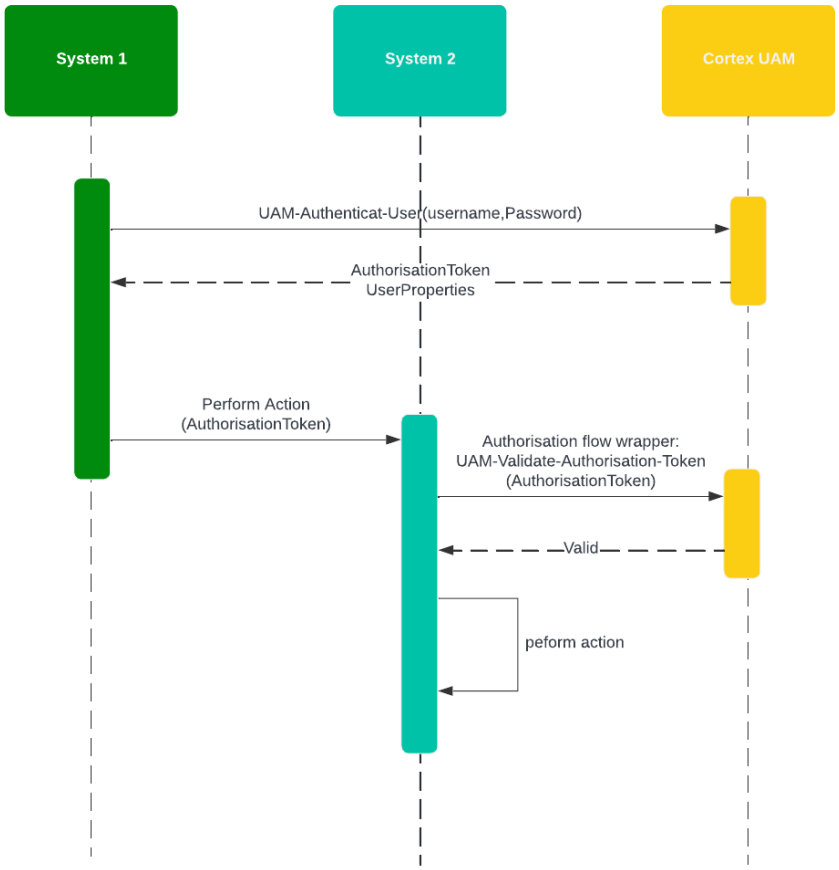


Figure 3 - UAM as authentication and authorisation between two systems

1.2 Implementation Recommendations

- **Login flow wrapper**

Although the module provides an out of the box authentication flow (UAM-Authenticate-User), it is recommended to build a project specific flow that wraps the module flow. This means you can build your own project flow will calls the user access management flow, and this allows you to manage and add additional features to the login flow without impacting other projects using the same module.

- **Authorisation flow wrapper**

Although the module provides an out of the box authorisation subtask (UAM-Create-User-Session), if the system requesting a token to be validated is not Cortex, it will be required to build a project specific flow that wraps the authorisation subtask flow. This allows you to manage and add additional features to the authorisation flow.

Examples of additional features that can be customized in the wrapper flows:

1. Authentication groups to application specific roles mapping

   When using Active Directory as the authentication system, this will output all groups that the user belongs to. It is recommended to make a translation layer between these groups and application specific roles so that these roles are abstracted, and unnecessary information is not passed back to the calling system. This also allows one active directory group to be mapped to several roles and facilitate the implementation of administration and super user roles.

2. Application specific authorisation mapping

   Depending on the solution, several applications could use the same authentication system but depending on the calling application, certain users should be restricted from logging in, regardless of having valid credentials. For these scenarios, it is recommended that the wrapper flow is built with an additional input, example i_Calling-Application, and a translation layer between calling applications and required roles is built.

   If for a specific application, the user does not belong to the required role, an exception can be thrown, and the session token invalidated.

3. Additional log information or security steps

   The use of a solution wrapper flow also allows specific logging and/or extra security steps to be built. These could be customised per solution without impacting the overall UAM module.

- **Credentials encryption**

It is advised that the username and password details are encrypted prior to calling the UAM-User-Authentication or the solution wrapper flow, to avoid any sensitive data to be logged during the flow execution. This should be done using the Cortex flow interface encrypt API.

2 User Access Management Flows

2.1 UAM-Authenticate-User

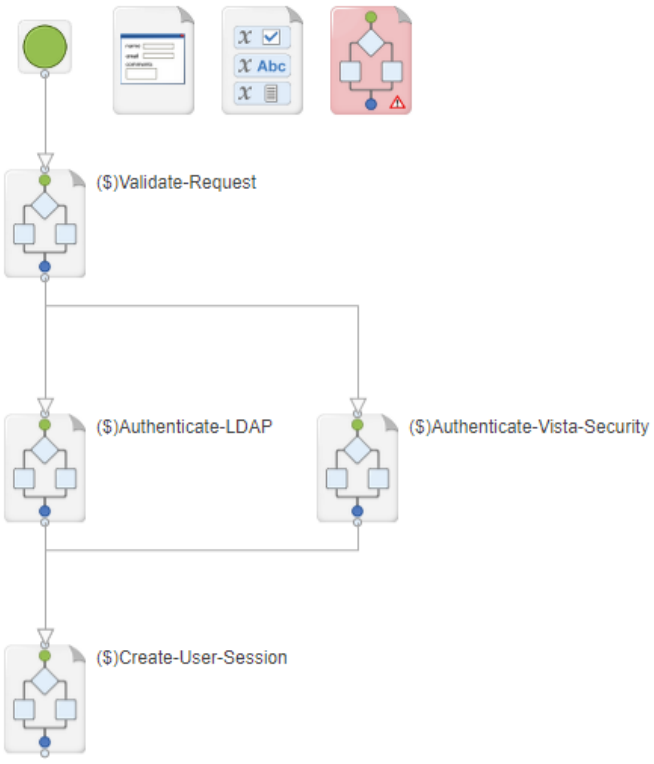
2.1.1 Overview

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.

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

2.1.3 Inputs

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

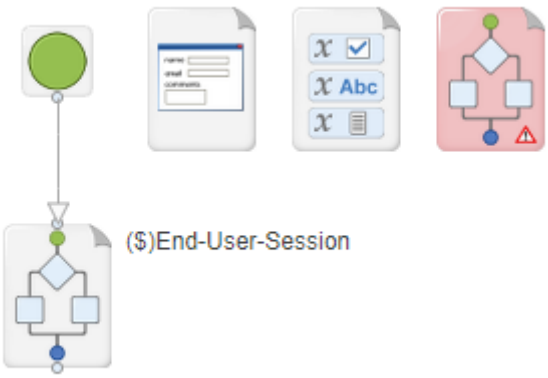
2.2.1 Overview

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

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

2.2.4 Outputs

2.3 UAM-Session-Management

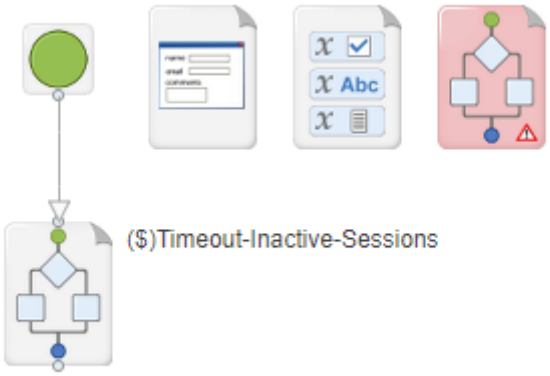
2.3.1 Overview

The UAM-Session-Management is used to timeout sessions that have been inactive for a set 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 set period of time, defined by the input variable i_Time-Out-Threshold.

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

2.3.3 Inputs

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

2.3.4 Outputs

-

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

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

3.1.2 Inputs

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

3.1.3 Outputs

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

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

3.2.2 Inputs

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

3.2.3 Outputs

4 Integration

4.1 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

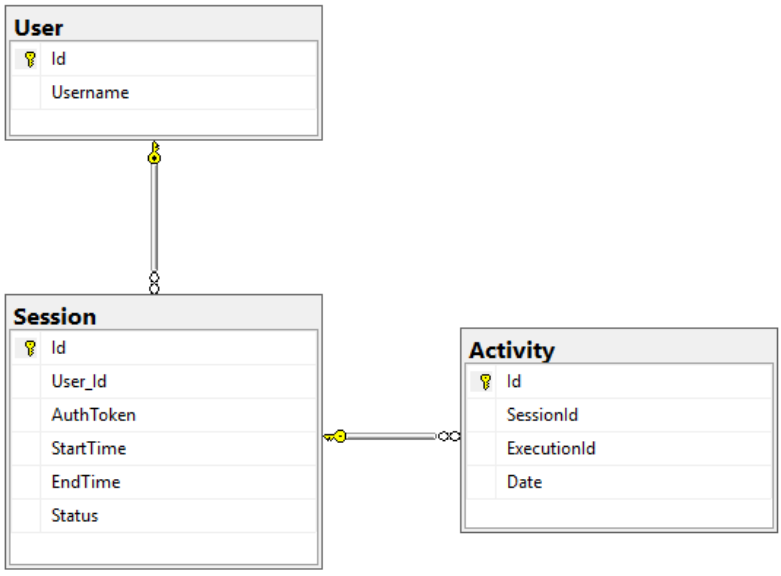


Figure 4 - Cortex User Access Management database schema

4.2 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'.

