Sertainty Self-Protecting-Data Technology

UXP >>> Unbreakable Exchange Protocol >>> UXP Technology

Data Life-Cycle Protection & Control

The UXP Technology Fit >>> Security at the Data Layer

UXP Technology is not an encryption tool. UXP Technology is not a security replacement.

UXP Technology is a built-in addition to your current security protocols.

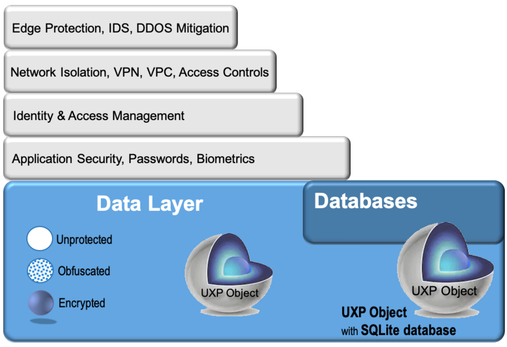


Figure 1. UXP Technology Fit: Security at the Data Layer

How UXP Technology is Different

>> UXP Technology Position: An Enhancement, Not a Replacement

UXP Technology positions itself as *an enhancement to the existing security layers*. UXP Technology enhances the existing security model by embedding intelligence and protection policies for access control into the data. From data's creation through its expiration or destruction, UXP Technology focuses on protecting the entire data life-cycle.

>> UXP Technology & the Data Life-Cycle

At the beginning of the data life-cycle, UXP Technology transforms your data into a UXP Object, an intelligent, proactive entity.

This transformation NEVER modifies the original data.

As a UXP Object, this entity is able to enforce embedded protection policies for access control wherever the Object resides. The protection policies, defined by the data owner, are suited for the data's security requirements.

During the data's active life-cycle, the embedded UXP Intelligence and protection policies empower the UXP Object to enforce the policies without deviation while the data is:

- In-Flight.
- In-Use.
- At-Rest.

The data owner stays in control of their data at all times.

This control remains in place during the UXP Object's entire life-cycle. The Object life-cycle is complete when one of the following events occur:

- Data is extracted or exported from the UXP Object to clear form.
- A specific protection policy is defined requiring one or all shown below:
    - Expiration date making the UXP Object inaccessible without shredding the content
    - UXP Object shreds the content based on these protocols:
        - Fixed policy expiration date for destruction
        - Mitigation policy trigger for destruction

>> UXP Protection Scheme

UXP Protection scheme is a unique protocol. UXP Technology uses proprietary algorithms that blend in obfuscation and cloaking along with standard AES 256-GCM encryption algorithms.

The overall UXP Protection scheme is embedded in the UXP Object.

The following components participate in the protection scheme generation:

- User Definitions (*Users permitted access to the UXP Object, inclusive of public and private attributes*)
- Protection policies for access control (*referred to as* Access and Mitigation Policies)
- Headers
- Metadata
- **Keys***
- Data (*optional*)

***Distinct Differentiator in the UXP Protection Scheme:** The protection protocol generates and embeds a key management system in the UXP Object. This system allows the UXP Object to manage the keys internally. Randomly generated when the data is protected, the embedded encryption keys are cloaked, inaccessible and never shared.

>> UXP Intelligence

UXP Technology's Intelligence solves the data security issue by embedding access controls, risk mitigations, and defensive mechanisms within the UXP Object. UXP Technology empowers the data through the UXP Object to be intelligent and proactive.

The actions listed below are samples of UXP Intelligence capabilities:

- User authentication
- Access policy enforcement
- Mitigation policy enforcement
- Real-time event/audit records
- Record of data ownership
- Embedded key management
- File-level redaction
- In-line read/write

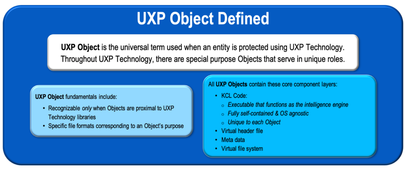
UXP Object

UXP Object

UXP Technology Cornerstone

The **UXP Object** is a portable protocol used to protect data in a self-managed, one-of-a-kind entity. UXP Technology blends proprietary UXP Intelligence and a unique protection scheme with any size dataset. The result is a UXP Object.

Empowered with UXP Technology, the UXP Object self-protects and self-governs its own access and mitigation activity. These activities are defined by the data-owner using the owner's pre-determined policies.



File Format

After a UXP Object is created, it appears as an inert, binary file showing a ***.uxp*** extension. The UXP Object is unidentifiable unless proximal to UXP Technology. Without proximity, a UXP Object simply looks ordinary and nondescript on any OS and can be easily designated as junk.

Internal Components

Beginning at a high level, the UXP Object contains the core components mentioned above along with keys and data:

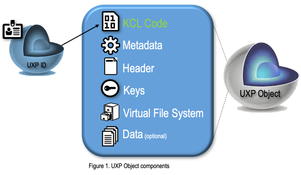


Figure 1. UXP Object components

The UXP Identity is a special purpose UXP Object that is required for UXP Object generation. As a pre-requisite, the UXP Identity provides necessary components that empower the UXP Object to be intelligent and proactive at an access attempt and while the data is in-use.

>> Virtual File System

The Virtual File System stores and manages within the UXP Object the following (including headers):

- KCL Code
- External data as files
- Internal metadata
- Event logs
- Other elements

Within the system, the files are segmented into pages. Each file is individually and uniquely protected with encryption and cloaking. Figure 2 below shows a simulation of an Object's Virtual File System.

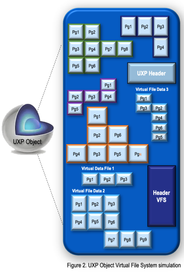


Figure 2. UXP Object Virtual File System simulation

Virtual File System supports a directory structure, individual files/folders or other data of any type.

Directory Structure: A directory structure can contain other virtual files as well as virtual sub-directories. This feature may require additional license privilege.

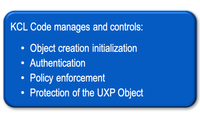
Data Type: Data can be any user-specified type, such as *.mp4, *.mp3, *.txt, etc. The virtual file can originate from either an existing file or an in-memory buffer. The file system can contain anything that normally goes into a conventional file system. The file system is only limited in size by the operating system itself.

On-Disk Storage Structures:

- Write Once UXP Architecture
  - A file system used for storing and protecting data that does not change, e.g., *.mp3 or*.mp4 files.
- Read/Write UXP Architecture
  - A file system used for storing data of any type that may change throughout its life cycle.
  - Database Tables *(available only on Read/Write)* are specific storage structures for storing and managing SQLite related database tables and files.

>> KCL Code: UXP Object Intelligence

The KCL Code originates from the UXP Identity. Acting as a light-weight executable embedded in the UXP Object, this UXP proprietary program is written in a C-like language; the KCL Code is the UXP Object's Intelligence.



Unique to each UXP Identity, the KCL Code consists of User Definitions and access and mitigation policies .

The KCL Code is also integral in the UXP Object's encryption key production. These keys are randomly generated by the UXP Engine, another proprietary engine external to the Object.

Additionally, the KCL Code manages all encryption keys internally throughout the Object's life cycle. Keys are embedded unseen in the Object and protected using a proprietary, recursive UXP protection scheme.

As an executable, the KCL Code requires proximity to UXP Technology, specifically the UXP Engine. The UXP Engine provides the executable environment for the KCL Code. Otherwise, the KCL Code sits dormant and undetectable in the inert UXP Object.

Note: For additional information on the KCL Code, see the KCL Guide.

>> Internal Metadata

The internal metadata contains all the virtual file system data. All other user-specific data is stored as virtual data files.

>> UXP Header

Every UXP Object contains a header that allows the UXP Engine to identify the entity as valid UXP Object. If a header isn't found or read, then the entity is not considered a valid UXP Object.

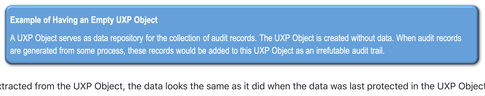
>> Encryption Keys

Standard AES-256 GCM encryption is involved in the UXP protection scheme. Added proprietary algorithms generate an embedded, self-contained key management system within the UXP Object. Generated when data is protected, encryption keys are cloaked, inaccessible and never shared.

The KCL Code participates in generating the encryption keys during UXP Object creation. Keys are randomly generated without exposure and contained hidden within the UXP Object. External key management becomes obsolete because the KCL Code manages the keys.

>> Data

Data can be any data type or size because data is treated as an array of bytes. Data, as shown in Figure 1 above, is optional within a UXP Object. An Object can be authenticated and opened to add data at a later time.



When the data is extracted from the UXP Object, the data looks the same as it did when the data was last protected in the UXP Object.

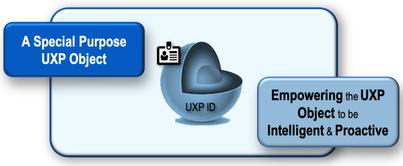
Tech Education Syllabus

UXP Identity

UXP Identity

Fundamental to UXP Object Creation

The UXP Identity plays a central role in UXP Technology. It is more than an username and password set; the UXP Identity extends beyond the traditional definition of a digital identity. As a portable, special purpose UXP Object, the UXP Identity is protected using the same the protection scheme as a UXP Object.



Function

The UXP Identity contains and protects the empowering components that permit the UXP Object to be intelligent and proactive. The UXP Identity continues securing the UXP Object, once authenticated, while the data is in-use.

The UXP Identity is necessary for generating a UXP Object. In a protected format, the Identity houses the owner specified access and mitigation policies and a list of permitted users for a dataset. These policies and the user list (*together, referred to as artifacts*) are required to create the unique protection scheme within the UXP Object. Before a UXP Object can be generated, a UXP Identity must be present at the time and place when a dataset is protected. It provides essential artifacts that are incorporated into the process for constructing the protection scheme for the UXP Object.

> ***Note:*** *Without the UXP Identity artifacts, the protection scheme can't be constructed, which ultimately prevents the UXP Object from being generated.*

File Format

Once created as file, the UXP Identity Object appears as an inert, binary file showing an ****.iic*** extension. This file type is unidentifiable unless proximal to UXP Technology libraries. It looks ordinary and nondescript on any O/S and can be easily designated as junk.

> ***Note:*** *No access parameters are seen or referred to in any way. These attributes along with additional unseen UXP Metadata provide verification details used by UXP Technology at the time when the UXP Identity is used to protect a dataset.*

Internal Components

The primary differences between a UXP Identity and a UXP Object are that the Identity is uneditable and contains no customer data. Additionally, the UXP Identity contains two types of artifacts, public and private. Both types are defined and controlled by the process or person who creates the UXP Identity.

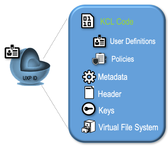


Figure 1. UXP Identity components

>> Virtual File System

The Virtual File System stores and manages within the UXP Identity the following (including headers):

- KCL Code
  - User Definitions
  - Policies
- Public artifacts
- Internal metadata
- Other elements

Within the system, the files are segmented into pages. Each file is individually and uniquely protected with encryption and cloaking. Figure 2 below shows a simulation of an Identity's Virtual File System.

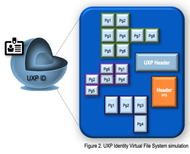


Figure 2. UXP Identity Virtual File System simulation

>> KCL Code

The KCL Code is created when the UXP Identity is created. Acting as a light-weight executable embedded in the UXP Identity, this UXP proprietary program is written in a C-like language. As shown in Figure 1 on the UXP Object page, the Identity provides the KCL Code for the Object. The KCL Code is the UXP Object's Intelligence.

KCL Code manages and controls:

- Object creation initialization
- Authentication
- Policy enforcement
- Protection of the UXP Object

Unique to each UXP Identity, the KCL Code consists of User Definitions and access and mitigation policies.

The KCL Code is also integral in the UXP Identity's encryption key production. The keys are randomly generated by the UXP Engine, another proprietary engine external to the Identity.

Additionally, the KCL Code manages all encryption keys internally throughout the UXP Identity's life cycle. Keys are embedded unseen in the Identity and protected using a proprietary, recursive UXP protection scheme.

As an executable, the KCL Code requires proximity to UXP Technology, specifically the UXP Engine. The UXP Engine provides the executable environment for the KCL Code. Otherwise, the KCL Code sits dormant and undetectable in the inert UXP Identity.

For additional information on the KCL Code, see the KCL Guide.

>>> User Definitions

User Definitions are embedded in the Identity and designated by the Identity owner. These User Definitions represent the static list of permitted users allow to access the UXP Object.

A User Definition includes:

- Valid User (*machine, process, human, or any combination*)
  - Challenge Pairs
    - Prompt
    - Response

> ***Note:*** *Prompts and Responses are used during an Object's access attempt. They are a 1:1 relationship. Each Prompt has a single corresponding Response; this 1:1 Prompt and Response relationship is referred to as a Challenge pair.*

- Optional device and location configurations, user specific

The User Definitions guide the KCL Code's intelligent and proactive actions during initial authentication.

User Definitions in the Developer Guide are referred to *User Credentials*. More specific information on User, see the Workflow Guide/Section 1.8 User Credentials.

>>> Policies

Policies are specific access and mitigation protocols defined by the data owner. Both access and mitigation policies center around behaviors designed to protect the UXP Object and its data. The KCL Code manages the policies within the UXP Object.

Policies are referred to as *rule presets* in throughout UXP Technology documentation. More specific information on the rule presets, see the Workflow Guide/Section 3.1.2 Rule Presets.

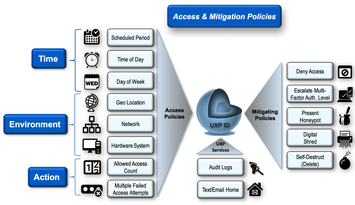


Figure 3. General overview of access and mitigation policies in the UXP Identity.

>> Internal Metadata

The Internal Metadata contains all the virtual file system data. All other user-specific data is stored as virtual data files.

>> UXP Header

Every UXP Identity contains a header that allows the UXP Engine to identify the entity as valid UXP Identity. If a UXP Header isn't found or read, then the entity is not considered a valid UXP Identity.

>> Encryption Keys

Standard AES-256 GCM encryption is involved in the UXP protection scheme, but the UXP scheme eliminates key sharing. The KCL Code participates in generating the encryption keys during UXP Identity creation. Keys are randomly generated without exposure and contained hidden within the UXP Identity. External key management becomes obsolete because the KCL Code manages the keys.

Tech Education Syllabus