I.Brief Introduction

The article describes the definitions, logic models and object Schema descriptions of Document Operation Language (UOML) based on unstructured data. Unstructured data cannot be abstracted to a three-layer model of structured data. For written documents of unstructured data whose logic depth possibly reach the tenth layer, we take the logic model as a reference to abstractly define a universal and representative UOML.

Keywords:

DocBase (Document Base), UOML (Unstructured-doucment Operation mark-up Language), PageSet, LayerSet

1. Audience

UOML users, UOML design reviewers, UOML Standardization Association, DocBase designers and DocBase developers

2. Overview

UOML is the operation interface orienting written information processing, which separates the core processing technologies (such as description, storage, processing, management and presentation) for written information from actual applications. The application model of UOML is implemented through the DocBase technology. According to the experiences from developing database systems, the implemented DocBase technologies divide industries in the field of written information processing, and forming the situation similar to database industry. The method has the following advantages:

1)Providing documents with the best versatility

Applications of different kinds access the DocBase through standard UOML, and extract document information through uniform interfaces, enabling the same document to be universal between different applications.

2)Forming industry division, reducing repeated development, and being more professional, complete and correct

Basic operations on documents are performed in the DocBase system, and an application needs not be developed repeatedly. Moreover, the DocBase system is developed by professional vendors, so specialty, completeness and

correctness of relevant technologies can be assured. Application software vendors and users can also choose the best DocBase system vendors to ensure the correctness and consistency of the processing effects.

3)Providing management mechanism for many documents (even mass documents)

Documents are organized efficiently for retrieving, querying and storage, as well as for embedding powerful information security mechanism.

4)Providing better security mechanism

Several roles can be set, and the permissions of each role can be set fine-granularly. Fine-granularity is duplex. On the one hand, you can set the permissions of the whole document or a tiny part of the document; on the other hand, you can set several types of permissions, not just the traditional read/write/inaccessible permissions.

5)Innovation and rational competition are encouraged

After the rational industry division is formed, DocBase system vendors and application vendors will compete with each other in corresponding areas. The condition, such as Microsoft Word mo0nopolized application software by the document format, will not occur. Each DocBase system vendor can also add new features beyond the standards to attract users, and the standards will not restrict innovation.

6)Optimizing performance with better portability and scalability

All platforms and performance comply with the same calling interface, thus you can continuously optimize performance and migrate it to different platforms without changing UOML standards.

3. Security Model

Overview

The information security model of the DocBase is role-oriented, consisting of access control and data signature.

Access control implements Discretionary Access Control (DAC), controlling the read, write and print permissions on node trees.

Data signature implements the signature of the node tree.

Permissions in access control and signatures in data signature are specified for roles.

Roles and Sessions

To access a security document, a session must be started. The data controlled by the permissions can only be accessed in the session, and signature and verification can only be implemented in the session.

An application can provide authentication mechanism in one session to log on as one or more roles. Permission checking, signature and verification are for the role as which current session logs on.

The definition of a role, which is identified by a pair of public and private keys, is valid DocBase-wide. The public keys of roles are saved in the DocBase, while the private keys of roles are saved outside of the DocBase. A front-end application is responsible for the security of roles. When you log on as a role in a session, you need to provide the private key of the role.

Discretionary Access Control (DAC)

Access control mechanism of the DocBase is discretionary access control (DAC), namely the security control mode allowing granting or revoking according to the current permissions.

This security model can control the access to a specified node and its sub-tree, which can be called as controlled node (tree). This model can control the read, write, print and administrative permissions of nodes. When authorizing or revoking permissions, current logged role must have administrative permission on corresponding node tree.

Some part of access control is implemented based on the public key encryption mechanism. The controlled node is encrypted in the storage layer, and the encrypted public and private keys are saved in the DocBase in the form of ciphertext. For role r, if r has read permission on the controlled node n, the DocBase must have saved the encrypted private key of n (encrypted using the public key of r); if r has write permission on n, the DocBase must have saved the encrypted public key of n (encrypted using the public key of r). This mechanism ensures that a role cannot get the plaintext of a node if the role does not have corresponding read and write permissions.

Data Signature

Data signature can sign the specified node sub-trees at any possible limited depth, from its root node.

Tree signature of the DocBase implements regularization at binary level. During tree signature, you must sort and code the structure and data of the tree in some way, and then sign the tree. Sorting and coding may cause trees with different data to form the same data to be signed. Currently implemented data signature mechanism can avoid same data caused by data sorting at tree structure, node and attribute levels.

At present, ECDSA is the supported signature mechanism.