Access Service API v1.5

Specifications

Disclaimer

PACOM Systems Pty Ltd makes no warranty of any kind with regard to this product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. PACOM Systems Pty Ltd shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this product. This document contains proprietary information and is protected by copyright. The information contained within this document is subject to change without notice. The PACOM website (www.pacom.com) contains the latest documentation updates.

Some options, compliance claims or procedures described herein may not be supported if old versions of device firmware and/or software are used.

Copyright notices

Compliance and accreditations

PACOM products comply with Advanced Encryption Standard (AES) FIPS 197 (encryption version 1.1).

Underwriters Laboratories Inc. (UL) and Intertek Electrical Testing Laboratories (ETL) are product safety standards/accreditors for North America. Product samples are tested to certain safety requirements, and periodic checks of manufacturers' facilities are carried out.

Software license notice

Your license agreement with PACOM Systems Pty Ltd, which is included with this product, specifies the permitted and prohibited uses of the product. It is protected by Australian and international copyright laws and international treaty obligations. Your rights to use the Software are limited by the terms stated below, and your use of the Software indicates your acceptance of these terms. If you do not agree with them, you must return, delete or destroy all copies of the Software.

Your rights to use the Software terminate immediately if you violate any of the following terms:

- Any unauthorized duplication or use in whole or in part, in print, or in any other storage and retrieval system is forbidden.
- You may not reverse-engineer, disassemble, decompile, or make any attempt to discover the source code of the Software.
- You may not modify the Software in any way whatsoever.

Trademarks

All trademarks, brand and product names are the property of their respective owners:

- Bouncy Castle (http://www.bouncycastle.org)
- #ziplib (http://www.icsharpcode.net/opensource/sharpziplib/)
- Mono Class Libraries (http://www.mono-project.com)
- NUnit (http://www.nunit.org)

PACOM Support

For product support, go to the PACOM (support.pacom.com).

Table of Contents

Overview

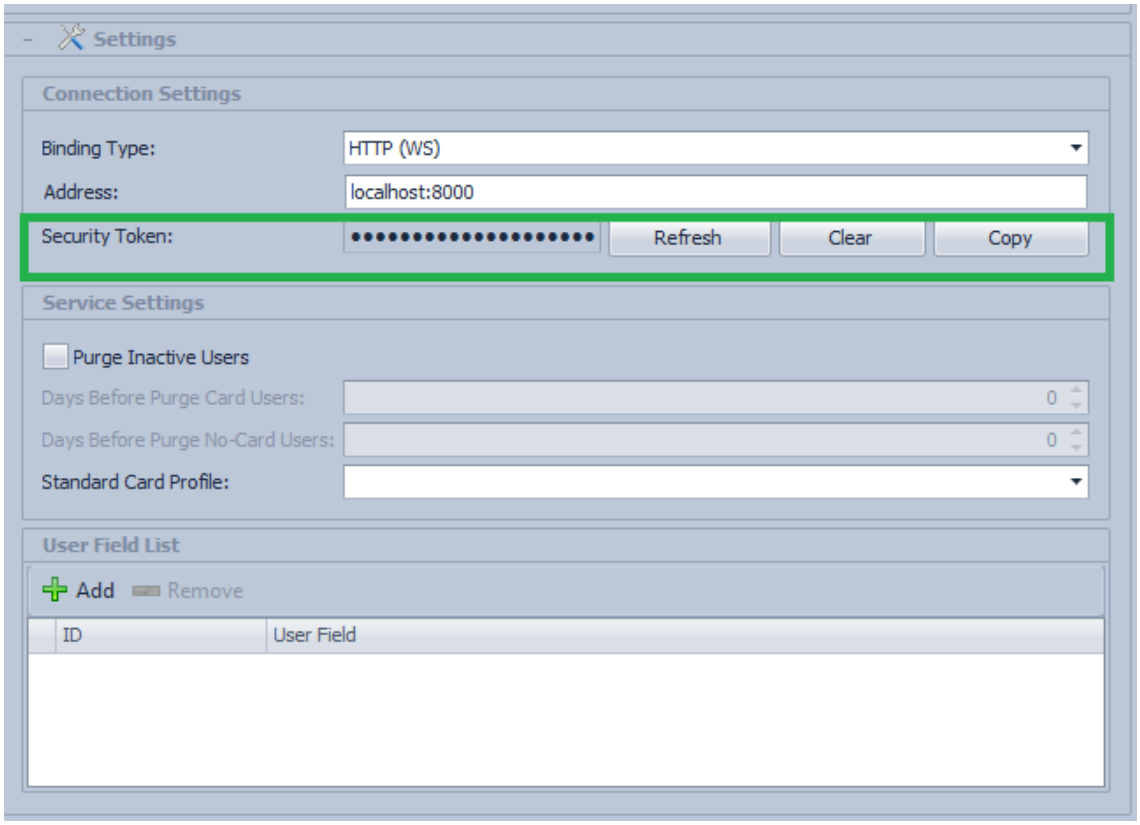
The Unison Access Service API enables an external system to read or modify the access control component of the Unison database using a simple web service interface. The web service type and address are specified in the Unison Access Service driver settings. By default, a standard HTTP (WS) service runs on `localhost:8000/Unison.AccessService` with a Metadata Exchange (WS-MetadataExchange) service running on `localhost:8000/Unison.AccessService/mex`.

Unison Access Service API v1.5 is compatible with Unison v5.10.

Updates

In Unison Access Service API v1.5, a token-based authentication has been added to the Access Service API driver.

A new field has been added to the Access Service Property page to enable token creation, refresh and clearance to be done manually by an operator / technician.



Access Service clients must request the security token to the operator or technician.

Available options:

- Refresh - create a new security token. The token is masked with asterisks (*).
- Clear - remove the contents of the **Security Token** field.
- Copy - Copy the value of the security token to the clipboard.

Note: The security token is only able to use HTTP/HTTPS binding types by adding a Unison - Token header in the HTTP request. The security header is not supported by other bindings.

For existing customers who do not wish to use any security tokens, the Access Service API will work as it currently does.

However, if customers generate tokens, the existing client must be modified to pass the Access Service API - Unison - Token: {token} in the header for every request. If for any reason, the token is refreshed, the client's connection is dropped and a new token must be created and re-connected.

Service API

The Unison Service API contains basic functionality for polling the service and getting version information from Unison.

Ping

Description	Used to check the connection. An exception is thrown if unable to connect.
Syntax	bool Ping()
Return Value	Type - bool True if Ping was successful.

SetTag

Description	Sets a tag. The tag is stored in the Unison database. It can be used to store any data.
Syntax	void SetTag(string tag)
Parameters	**string** *tag* Any text.

GetTag

Description	Gets the tag.
Syntax	string GetTag()
Return Value	Type - string The string value set with SetTag.

GetVersion

Description	Retrieves information about the current API and Unison versions.
Syntax	ServiceVersion GetVersion()
Return Value	Type - ServiceVersion Data contract with version numbers.

Import API

The Unison Import API enables card access data such as users and access groups from an external system to be imported into Unison.

UpdateCard

Description	Creates or updates a card. The numbers together uniquely identify the card. An exception is thrown if it is not possible to create the card.	
Syntax	void UpdateCard(string userId, string profileName, string cardNumber, string systemNumber, string versionNumber, string miscNumber, CardStatus cardStatus)	
Parameters	string userId	The string representing the unique ID of this user in the database. Only needed if creating a new card.
	string profileName	The Unison card profile to use. The default card profile (set in the hardware settings of the driver) is used if this parameter is null or empty.
	string cardNumber	The primary number of the card in the format specified by the card profile.
	string systemNumber	The system number of the card (if used) in the format specified by the card profile.
	string versionNumber	The version number of the card (if used) in the format specified by the card profile.
	string miscNumber	The miscellaneous number of the card (if used) in the format specified by the card profile.
	CardStatus cardStatus	Sets the card status. Set to Active if a new card is created and this parameter is NoChange.

UpdateCardByKey

Description	Creates or updates a card by key. An exception is thrown if it is not possible to create the card.	
Syntax	int UpdateCardByKey(int cardKey, int userKey, int profileKey, string cardNumber, string systemNumber, string versionNumber, string miscNumber, CardStatus cardStatus)	
Parameters	int cardkey	The unique database key for the card. Use the value 0 (zero) to create a new card.
	int userKey	The unique database key for the user. Only needed if creating a new card.
	int profileKey	The unique database key of the Unison card profile to use. The default card profile (set in the hardware settings of the driver) is used if this parameter is 0 (zero).
	string cardNumber	The primary number of the card in the format specified by the card profile.
	string systemNumber	The system number of the card (if used) in the format specified by the card profile.
	string versionNumber	The version number of the card (if used) in the format specified by the card profile.
	string miscNumber	The miscellaneous number of the card (if used) in the format specified by the card profile.
	CardStatus cardStatus	Sets the card status. Set to Active if a new card is created and this parameter is NoChange.
Return Value	Type - int The database key of the created or updated card.	

UpdateBinaryCard

Description	Creates or updates a card. The numbers together uniquely identify the card. An exception is thrown if it is not possible to create the card.	
Syntax	void UpdateBinaryCard(string userId, string profileName, byte[] cardNumber, byte[] systemNumber, byte[] versionNumber, byte[] miscNumber, CardStatus cardStatus)	
Parameters	string userId	The string representing the unique ID of the user in the database. Only needed if creating a new card.
	string profileName	The Unison card profile to use. The default card profile (set in the hardware settings of the driver) is used if this parameter is null or empty.
	byte[] cardNumber	The primary number of the card in raw binary format.
	byte[] systemNumber	The system number of the card (if used) in raw binary format.
	byte[] versionNumber	The version number of the card (if used) in raw binary format.
	byte[] miscNumber	The miscellaneous number of the card (if used) in raw binary format.
	CardStatus cardStatus	Sets the card status. Set to Active if a new card is created and this parameter is NoChange.

UpdateBinaryCardByKey

Description	Creates or updates a card. The numbers together uniquely identify the card. An exception is thrown if it is not possible to create the card.	
Syntax	int UpdateBinaryCardByKey(int cardKey, int userKey, int profileKey, byte[] cardNumber, byte[] systemNumber, byte[] versionNumber, byte[] miscNumber, CardStatus cardStatus)	
Parameters	int cardkey	The unique database key for the card. Use the value 0 (zero) to create a new card.
	int userKey	The unique database key for the user. Only needed if creating a new card.
	int profileKey	The unique database key of the Unison card profile to use. The default card profile (set in the hardware settings of the driver) is used if this parameter is 0 (zero).
	byte[] cardNumber	The primary number of the card in raw binary format.
	byte[] systemNumber	The system number of the card (if used) in raw binary format.
	byte[] versionNumber	The version number of the card (if used) in raw binary format.
	byte[] miscNumber	The miscellaneous number of the card (if used) in raw binary format.
	CardStatus cardStatus	Sets the card status. Set to Active if a new card is created and this parameter is NoChange.
Return Value	Type- int The database key of the created or updated card.	

RemoveCard

Description	Remove a card for a user. The numbers together uniquely identify the card.	
Syntax	void RemoveCard(string userId, string profileName, string cardNumber, string systemNumber, string versionNumber, string miscNumber)	
Parameters	string userId	The string representing the unique ID of the user in the database.
	string profileName	The Unison card profile to use. The standard card profile (set in the Unison Access Service driver) is used if this parameter is null or empty.
	string cardNumber	The primary number of the card in the format specified by the card profile.
	string systemNumber	The system number of the card (if used) in the format specified by the card-profile.
	string versionNumber	The version number of the card (if used) in the format specified by the card-profile.
	string miscNumber	The miscellaneous number of the card (if used) in the format specified by the card profile.

RemoveCardByKey

Description	Remove a card for a user. The numbers together uniquely identify the card.	
Syntax	void RemoveCardByKey(int cardKey)	
Parameters	int cardKey	The unique database key for the card.

UpdateUser

Description	Creates or updates a user. The `userId` field must always be set and match exactly. If a user with this `userId` cannot be found, a user is created.	
Syntax	void UpdateUser(string userId, string firstName, string lastName, string pinCode, DateTime? validFrom, DateTime? validUntil, AccessFlags accessFlags, List<UserField> fields)	
Parameters	string userId	The string representing the unique ID of this user in the database.
	string firstName	The given name of the user. The field in the database will not be changed if this parameter is null or empty.
	string lastName	The surname of the user. The field in the database will not be changed if this parameter is null or empty.
	string pinCode	The PIN code of the user. The code should be the correct length and padded with zeros. The field in the database will not be changed if this parameter is null or empty. A random PIN code will be created if a new user is created and this parameter is null.
	DateTime validFrom	The valid from date of the user. A date less than January 1, 1753 will be parsed as limitless. The field in the database will not be changed if this parameter is null. Today's date will be used if a new user is created and this parameter is null.
	DateTime validUntil	The valid until date of the user. A date greater than December 31, 9999 will be parsed as limitless. The field in the database will not be changed if this parameter is null. No limit will be set if a new user is created and this parameter is null.
	AccessFlags accessFlags	The access flags for the user that specifies special system operations. The field in the database will not be changed if this parameter has the NoChange flag set.
	List<UserField> fields	The dynamic fields in the database for this user. Missing fields will not be changed.

UpdateUserByKey

Description	Creates or updates a user using the database key. The `userKey` field must always be set and match the unique database key for the user.	
Syntax	int UpdateUserByKey(int userKey, string userId, string firstName, string lastName, string pinCode, DateTime? validFrom, DateTime? validUntil, AccessFlags accessFlags, List<UserField> fields)	
Parameters	int userKey	The unique database key for this user. Use the value 0 (zero) to create a new user.
	string userId	The string representing the unique ID of this user in the database. The field in the database will not be changed if this parameter is null or empty.
	string firstName	The given name of the user. The field in the database will not be changed if this parameter is null or empty.
	string lastName	The surname of the user. The field in the database will not be changed if this parameter is null or empty.
	string pinCode	The PIN code of the user. The code should be the correct length and padded with zeros. The field in the database will not be changed if this parameter is null or empty. A random PIN code will be created if a new user is created and this parameter is null.
	DateTime validFrom	The valid from date of the user. A date less than January 1, 1753 will be parsed as limitless. The field in the database will not be changed if this parameter is null. Today's date will be used if a new user is created and this parameter is null.
	DateTime validUntil	The valid until date of the user. A date greater than December 31, 9999 will be parsed as limitless. The field in the database will not be changed if this parameter is null. No limit will be set if a new user is created and this parameter is null.
	AccessFlags accessFlags	The access flags for the user that specifies special system operations. The field in the database will not be changed if this parameter has the NoChange flag set.
	List<UserField> fields	The dynamic fields in the database for this user. Missing fields will not be changed.

Return Value	Type - int The database key of the created or updated user.

RemoveUser

Description	Removes a user from the database. The `userId` must always be set and match exactly. An argument exception is thrown if no user with this `userId` can be found.	
Syntax	void RemoveUser(string userId)	
Parameters	string userId	The string representing the unique ID of this user in the database.

RemoveUserByKey

Description	Removes a user from the database. The `userId` must always be set and match exactly. An argument exception is thrown if no user with this `userId` can be found.	
Syntax	void RemoveUserByKey(int userKey)	
Parameters	int userKey	The unique database key for this user.

ChangeUserId

Description	Changes the `userId` of a user. An a rgument exception is thrown if no user with this `oldUserId` can be found. Use UpdateUserByKey() to change the `userId` when using key based lookup.	
Syntax	void ChangeUserId(string oldUserId, string newUserId)	
Parameters	string oldUserId	The current `userId` of the user.
	string newUserId	The new `userId` of the user.

UpdateMembership

Description	Adds or updates an access group membership for a user. An argument exception ise thrown if no user with this `userId` can be found or if no access group with the correct name can be found.	
Syntax	void UpdateMembership(string userId, string groupName, DateTime? validFrom, DateTime? validUntil)	
Parameters	string userId	The string representing the unique ID of this user in the database.
	string groupName	The name of the access group in Unison. The string must match exactly.
	DateTime validFrom	The valid from date of the membership. A date less than January 1, 1753 will be parsed as limitless. The field in the database will not be changed if this parameter is null. Today's date will be used if a new membership is created and this parameter is null.
	DateTime validUntil	The valid until date of the membership. A date greater than December 31, 9999 will be parsed as limitless. The field in the database will not be changed if this parameter is null. No limit will be set if a new membership is created and this parameter is null.

UpdateMembershipByKey

Description	Adds or updates an access group membership for a user. An argument exception is thrown if no user with this `userId` can be found or if no access group with the correct name can be found.	
Syntax	int UpdateMembershipByKey(int membershipKey, int userKey, int groupKey, DateTime? validFrom, DateTime? validUntil)	
Parameters	int membershipKey	The unique database key for this membership. Use the value 0 (zero) to create a new membership (or to lookup based on `userKey` and `groupKey` instead).
	int userKey	The unique database key for the user. It can be zero if using `membershipKey` to update an existing membership.
	int groupKey	The unique database key for the access group. It can be zero if using `membershipKey` to update an existing membership.
	DateTime validFrom	The valid from date of the membership. A date less than January 1, 1753 will be parsed as limitless. The field in the database will not be changed if this parameter is null. Today's date will be used if a new membership is created and this parameter is null.
	DateTime validUntil	The valid until date of the membership. A date greater than December 31, 9999 will be parsed as limitless. The field in the database will not be changed if this parameter is null. No limit will be set if a new membership is created and this parameter is null.
Return Value	Type - int The database key of the created or updated membership.	

RemoveMembership

Description	Removes an access group membership for a user.	
Syntax	void RemoveMembership(string userId, string groupName)	
Parameters	string userId	The string representing the unique ID of the user in the database.
	string groupName	The name of the access group in Unison. This string must match exactly.

RemoveMembershipByKey

Description	Removes an access group membership for a user.	
Syntax	void RemoveMembershipByKey(int membershipKey, int userKey, int groupKey)	
Parameters	int membershipKey	The unique database key for this membership. Use the value 0 (zero) to lookup based on `userKey` and `groupKey` instead.
	int userKey	The unique database key for the user. It can be zero if using `membershipKey` to remove an existing membership.
	int groupKey	The unique database key for the access group. It can be zero if using `membershipKey` to remove an existing membership.

UpdateUserPhoto

Description	Sets photo for user. An exception is thrown if the user is not found.	
Syntax	void UpdateUserPhoto(string userId, byte[] photo)	
Parameters	string userId	The string representing the unique ID of the user in the database.
	byte[] photo	Byte array representing a jpeg image.

UpdateUserPhotoByUserKey

Description	Sets photo for user. An exception is thrown if the user is not found.	
Syntax	void UpdateUserPhotoByUserKey(int userKey, byte[] photo)	
Parameters	int userKey	Unique database key for user.
	byte[] photo	Byte array representing a jpeg image.

UpdateUserSignature

Description	Sets a signature for the user. An exception is thrown if the user is not found.	
Syntax	void UpdateUserSignature(string userId, byte[] signature)	
Parameters	string userId	The string representing the unique ID of the user in the database.
	byte[] signature	Byte array representing a jpeg image.

UpdateUserSignatureByUserKey

Description	Sets a signature for the user. An exception is thrown if the user is not found.	
Syntax	void UpdateUserSignatureByUserKey(int userKey, byte[] signature)	
Parameters	int userKey	Unique database key for user.
	byte[] signature	Byte array representing a jpeg image.

ParseText

Description	Parses different kinds of pre-specified import data texts.	
Syntax	int ParseText(TextFormat textFormat, string text)	
Parameters	TextFormat textFormat	Specifies the format of the parsed text.
	string text	The text to parse.
Return Value	Type - int 0 if successfully parsed -1 for errors	

Export API

The Unison Export API enables card access data such as users and access groups to be exported to an external system from Unison.

GetAllUsers

Description	Retrieves Unison users.	
Syntax	List<UserInfo> GetAllUsers(int fromKey, int maxCount)	
Parameters	int fromKey	Only users who have a higher key in the Unison database are retrieved.
	int maxCount	The maximum number of users retrieved.
Return Value	Type - List<UserInfo> A list of user information.	

GetUserByKey

Description	Retrieves a user.	
Syntax	UserInfo GetUserByKey(int key)	
Parameters	int key	A key that identifies the user in the Unison database.
Return Value	Type - UserInfo User information or null if no user is found.	

GetUserById

Description	Retrieves a user with a specified user ID.	
Syntax	UserInfo GetUserById(string idy)	
Parameters	string id	A string representing the unique ID in the Unison database.
Return Value	Type - UserInfo User information or null if no user is found.	

GetAllUserFields

Description	Retrieves user fields from the Unison database.	
Syntax	List<UserFieldInfo> GetAllUserFields(int fromKey, int maxCount)	
Parameters	int fromKey	Only user fields with higher value for the unique key will be retrieved.
	int maxCount	The maximum number of user fields that are retrieved.
Return Value	Type - List<UserFieldInfo> A list of user field information.	

GetUserFieldsByUserKey

Description	Retrieves user fields for a specified user.	
Syntax	List<UserFieldInfo> GetUserFieldsByUserKey(int userKey)	
Parameters	int userKey	The unique key in the Unison database that specifies a user.
Return Value	Type - List<UserFieldInfo> A list of user field information.	

GetAllUserFieldIds

Description	Retrieves unique keys from the Unison database for user fields.
Syntax	List<UserFieldIdInfo> GetAllUserFieldIds()
Return Value	Type - List<UserFieldIdInfo> A list of user field ID information.

GetAllAccessGroups

Description	Retrieves access groups from the Unison database.	
Syntax	List<AccessGroupInfo> GetAllAccessGroups(int fromKey, int maxCount)	
Parameters	int fromKey	Access groups must have a higher value for a unique identifier to be found.
	int maxCount	The maximum number of access groups that are retrieved.
Return Value	Type - List<AccessGroupInfo> A list of access group information.	

GetAccessGroupByKey

Description	Retrieves access groups with a specified key.	
Syntax	AccessGroupInfo GetAccessGroupByKey(int key)	
Parameters	int key	A unique key in the Unison database that defines the access group.
Return Value	Type - AccessGroupInfo Access group information for found group or null if an access group is not found.	

GetAccessGroupByName

Description	Retrieves access groups with a specified name.	
Syntax	AccessGroupInfo GetAccessGroupByName(string name)	
Parameters	string name	The name of an access group in the Unison database.
Return Value	Type - AccessGroupInfo Access group information for found group or null if an access group is not found.	

GetAllMemberships

Description	Retrieves memberships from the Unison database.	
Syntax	List<MembershipInfo> GetAllMemberships(int fromKey, int maxCount)	
Parameters	int fromKey	Memberships must have a higher value for the unique key to be found.
	int maxCount	The maximum number of memberships that are retrieved.
Return Value	Type - List<MembershipInfo> A list of membership information.	

GetMembershipsByUserKey

Description	Locates memberships in the Unison database for a specified user.	
Syntax	List<MembershipInfo> GetMembershipsByUserKey(int userKey)	
Parameters	int userKey	A unique key for the user in the Unison database.
Return Value	Type - List<MembershipInfo> A list of membership information for the user.	

GetMembershipsByAccessGroupKey

Description	Locates memberships in the Unison database for a specified access group.	
Syntax	List<MembershipInfo> GetMembershipsByAccessGroupKey(int groupKey)	
Parameters	int groupKey	A unique key for the access group in the Unison database.
Return Value	Type - List<MembershipInfo> A list of membership information that references the access group.	

GetAllCards

Description	Locates access cards in the Unison database.	
Syntax	List<CardInfo> GetAllCards(int fromKey, int maxCount)	
Parameters	int fromKey	Cards must have a higher value for the unique key to be found.
	int maxCount	The maximum number of cards that are retrieved.
Return Value	Type - List<CardInfo> A list of card information of found cards.	

GetCardsByUserKey

Description	Locates cards that are owned by specified user.	
Syntax	List<CardInfo> GetCardsByUserKey(int userKey)	
Parameters	int userKey	A unique key for the card owner in the Unison database.
Return Value	Type - List<CardInfo> A list of card information of found cards.	

GetCardByKey

Description	Locates an access card.	
Syntax	CardInfo GetCardByKey(int key)	
Parameters	int key	A unique key for the card in the Unison database.
Return Value	Type - List<CardInfo> A list of card information or null if no card is found.	

GetCardByNumber

Description	Locates an access card with specified card data. Values for fields that are not used for the card profile should be empty strings.	
Syntax	CardInfo GetCardByNumber(string cardNumber, string systemNumber, string versionNumber, string miscNumber)	
Parameters	string cardNumber	The card number.
	string systemNumber	The system number.
	string versionNumber	The version number.
	string miscNumber	The miscellaneous number.
Return Value	Type - CardInfo A list of card information or null if no card is found.	

GetAllCardProfiles

Description	Retrieves card profiles from the Unison database.
Syntax	List<CardProfileInfo> GetAllCardProfiles()
Return Value	Type - CardProfileInfo Information about the card profiles.

GetUserImagesByUserKey

Description	Retrieves user images, that is, photos and signatures.	
Syntax	UserImagesInfo GetUserImagesByUserKey(int userKey)	
Parameters	int userKey	A unique key that defines the user in the Unison database for which images are located.
Return Value	Type - UserImagesInfo Information about a user's images (that is, photos and signatures).	

Export Synchronization API

The Unison Export Synchronization API enables synchronization of user and access group data from Unison to an external system when changes in data occur in Unison.

SyncReset

Description	Forces a reset of the synchronization.
Syntax	void SyncReset()

SyncBegin

Description	Retrieves synchronization status and updates synchronization versions for users, cards, user fields, memberships, access groups and user images. Synchronization versions are used for identifying when data has been changed and synchronization is necessary. This method must be called at the start of synchronization.
Syntax	SyncStatus SyncBegin()
Return Value	Type - SyncStatus Returns data types that are not up-to-date. The result is a merged enumerator.

SyncEnd

Description	Stores synchronization status. This method should be called after synchronization is complete.
Syntax	void SyncEnd()

SyncUsers

Description	Starts the synchronization of users.
Syntax	List<UserInfo> SyncUsers(int maxCount)
Parameters	int maxCount The maximum number of users to synchronize.
Return Value	Type - List<UserInfo> A list of changed users.

SyncUserFields

Description	Starts the synchronization of user fields. User fields are custom user data fields that can be dynamically added or removed in Unison.
Syntax	List<UserFieldInfo> SyncUserFields(int maxCount)
Parameters	int maxCount The maximum number of user fields to synchronize.
Return Value	Type - List<UserFieldInfo> A list of retrieved user field information.

SyncAccessGroups

Description	Starts the synchronization of access groups.
Syntax	List<AccessGroupInfo> SyncAccessGroups(int maxCount)
Parameters	int maxCount The maximum number of access groups to synchronize.
Return Value	Type - List<AccessGroupInfo> A list of retrieved access group information.

SyncMemberships

Description	Starts the synchronization of memberships. A membership defines which access group a user has access to.
Syntax	List<MembershipInfo> SyncMemberships(int maxCount)
Parameters	int maxCount The maximum number of retrieved memberships.
Return Value	Type - List<MembershipInfo> A list of retrieved membership information.

SyncCards

Description	Starts the synchronization of user access cards.
Syntax	List<CardInfo> SyncCards(int maxCount)
Parameters	int maxCount The maximum number of retrieved cards.
Return Value	Type - List<CardInfo> A list of retrieved card information.

SyncUserImages

Description	Starts the synchronization of user images, that is, photos and signatures.
Syntax	List<UserImagesInfo> SyncUserImages(int maxCount)
Parameters	int maxCount The maximum number of images to synchronize.
Return Value	Type - List<UserImagesInfo> A list of retrieved user images information.

AccessFlags

Description	A flag enumeration representing the user flags to perform specific commands.	

Syntax	`[Flags]`
	`enum AccessFlags`
	`{`
	`None`
	`AllowArm`
	`AllowDisarm`
	`ExtendedTimes`
	`CommandControl`
	`NoChange`
	`}`	

Values	None	No user flag is set.
	AllowArm	Allows users to arm alarm areas.
	AllowDisarm	Allows users to disarm alarm areas.
	ExtendedTimes	User has extended time when accessing a door.
	CommandControl	Allows users to perform specific commands at the reader.
	NoChange	No change to the user flags is to be done.

AccessGroupInfo

Description	A class representing the access group information.	

Syntax	`[DataContract]`
	`class AccessGroupInfo : BaseInfo`
	`{`
	`int Key`
	`string Name`
	`}`	

Properties	int Key	The access group database key.
	string Name	The name of the access group.

ArgumentError

Description	A class representing an argument error.

Syntax	`[DataContract]`
	`class ArgumentError`
	`{`
	`string Message`
	`}`

Properties	string Message — The argument error message string.

BaseInfo

Description	Base class for information fields.

Syntax	`[DataContract]`
	`class BaseInfo`
	`{`
	`long Version`
	`bool IsDeleted`
	`}`

Properties	long Version — The current version of the field.
	bool IsDeleted — Indicationif the field is deleted.

CardInfo

Description	A class representing the user card information.

Syntax	`[DataContract]`
	`class CardInfo : BaseInfo`
	`{`
	`int Key`
	`int UserKey`
	`int ProfileKey`
	`int? LayoutKey`
	`string CardNumber`
	`string SystemNumber`
	`string VersionNumber`
	`string MiscNumber`
	`byte[] BinaryCardNumber`
	`byte[] BinarySystemNumber`
	`byte[] BinaryVersionNumber`
	`byte[] BinaryMiscNumber`
	`byte[] BinaryCardData`
	`CardStatus Status`
	`}`

Properties	int Key	The user card database key.
	int UserKey	The user database key.
	int ProfileKey	The user card profile database key.
	int? LayoutKey	The user card layout database key.
	string CardNumber	The user's card number.
	string SystemNumber	The user's card system number.
	string VersionNumber	The user's card version number.
	string MiscNumber	The user's card miscellaneous number.
	byte[] BinaryCardNumber	The user's card number in binary format.
	byte[] BinarySystemNumber	The user's card system number in binary format.
	byte[] BinaryVersionNumber	The user's card version number in binary format.
	byte[] BinaryMiscNumber	The user's card miscellaneous number in binary format.
	byte[] BinaryCardData	The user's card complete card data in binary format.
	CardStatus Status	The user's card status.

CardProfileInfo

Description	A class representing the card profile information.	

Syntax	`[DataContract]`
	`class CardProfileInfo : BaseInfo`
	`{`
	`int Key`
	`string Name`
	`}`	

Properties	int Key	The card profile database key.
	string Name	The name of the card profile.

CardStatus

Description	An enumeration representing the user card status.	

Syntax	`enum CardStatus`
	`{`
	`NoChange`
	`Active`
	`Blocked`
	`Lost`
	`Canceled`
	`}`	

Values	NoChange	No status change.
	Active	The card is marked as active.
	Blocked	The card is marked as blocked.
	Lost	The card is marked as lost.
	Canceled	The card is marked as canceled.

FieldAction

Description	An enumeration representing the actions to perform for the User Fields.	

Syntax	`enum FieldAction`
	`{`
	`SetValue`
	`AddValue`
	`RemoveValue`
	`Unknown`
	`}`	

Values	SetValue	Sets the field to the specified value.
	AddValue	Adds the specified value to the end of the field.
	RemoveValue	Removes a value added with `AddValue` anywhere in the field.
	Unknown	Placeholder for unknown action.

MembershipInfo

Description	A class representing the user access group membership information.	

Syntax	`[DataContract]`
	`class MembershipInfo : BaseInfo`
	`{`
	`int Key`
	`int UserKey`
	`int AccessGroupKey`
	`DateTime ValidFrom`
	`DateTime ValidUntil`
	`}`	

Properties	int Key	The access group membership database key.
	int UserKey	The user database key.
	int AccessGroupKey	The access group database key.
	DateTime ValidFrom	The date and time the access group membership is valid from.
	DateTime ValidUntil	The date and time the access group membership is valid until.

ServiceError

Description	A class representing a service error.	

Syntax	`[DataContract]`
	`class ServiceError`
	`{`
	`string Message`
	`string Body`
	`}`	

Properties	string Message	The service error message string.
	string Body	The service error message body.

ServiceVersion

Description	A class representing the current version of Unison.

Syntax	`[DataContract]`
	`class ServiceVersion`
	`{`
	`int APIMajorVersion`
	`int APIMinorVersion`
	`string UnisonVersion`
	`int UnisonBuild`
	`}`

Properties	int APIMajorVersion — The major version of the Unison Access Service API.
	int APIMinorVersion — The minor version of the Unison Access Service API.
	string UnisonVersion — The Unison software version.
	int UnisonBuild — The Unison software build version.

SyncResetNeededError

Description	A class representing a synchronization reset needed error.

Syntax	`[DataContract]`
	`class SyncResetNeededError`
	`{`
	`string Message`
	`}`

Properties	string Message — The synchronization reset needed error message string.

SyncStatus

Description	A flag enumeration representing the synchronization status.

Syntax	`[Flags]`
	`enum SyncStatus`
	`{`
	`None`
	`UsersDirty`
	`UserFieldsDirty`
	`MembershipsDirty`
	`AccessGroupsDirty`
	`CardsDirty`
	`UserImagesDirty`
	`AllDirty`
	`FirstBatch`
	`}`

Values	None	No syncronization has been performed.
	UsersDirty	Syncronization for users is required.
	UserFieldsDirty	Syncronization for user fields is required.
	MembershipsDirty	Syncronization for user access group memberships is required.
	AccessGroupsDirty	Syncronization for access groups is required.
	CardsDirty	Syncronization for cards is required.
	UserImagesDirty	Syncronization for user images is required.
	AllDirty	Syncronization for all data is required.
	FirstBatch	Indicates the first syncronization batch.

TextFormat

Description	An enumeration representing the text format for the ParseText function.	

Syntax	`enum TextFormat`
	`{`
	`GUSampassXML`
	`}`	

Values	GUSampassXML	Text format is GUSampassXML (custom format).

UserField

Description	A class representing a UserField change action.	

Syntax	`[DataContract]`
	`class UserField`
	`{`
	`public int Id`
	`public FieldAction Action`
	`public string Value`
	`}`	

Properties	int Id	ID of the field. The field ID mapping is set up in the Unison Access Service driver.
	FieldAction Action	The action to perform on the field.
	string Value	The value to use for the requested action.

UserFieldIdInfo

Description	A class representing the user customer fields ID information.	

Syntax	`[DataContract]`
	`class UserFieldIdInfo : BaseInfo`
	`{`
	`int Id`
	`string Name`
	`UserFieldIdType IdType`
	`string RegExp`
	`bool IsUnique`
	`List<string> PossibleValues`
	`}`	

Properties	int Id	The user field ID.
	string Name	The user field name.
	UserFieldIdType IdType	The user field ID type.
	string RegExp	The regular expression for the user field.
	bool IsUnique	Indicates if the user field is unique.
	List<string> PossibleVlaues	A list of possible values for the user field.

UserFieldIdType

Description	An enumeration representing custom user fields ID type.	

Syntax	`enum UserFieldIdType`
	`{`
	`RegExp`
	`List`
	`EditList`
	`}`	

Values	RegExp	Custom field is masked with a regular expression.
	List	Custom field is a list field.
	EditList	Custom field is an edit list field.

UserFieldInfo

Description	A class representing the user customer fields information.	

Syntax	`[DataContract]`
	`class UserFieldInfo : BaseInfo`
	`{`
	`int Key`
	`int Id`
	`int UserKey`
	`string Value`
	`}`	

Properties	int Key	The user field database key.
	int Id	The user field ID.
	int UserKey	The user's database key.
	string Value	The value of the field.

UserImagesInfo

Description	A class representing the user images information, that is, photos and signatures.	

Syntax	`[DataContract]`
	`class UserImagesInfo : BaseInfo`
	`{`
	`int Key`
	`int UserKey`
	`byte[] Photo`
	`byte[] Signature`
	`}`	

Properties	int Key	The user's images database key.
	int UserKey	The user's database key.
	byte[] Photo	The user's photo in binary jpeg format.
	byte[] Signature	The user's signature in binary jpeg format.

UserInfo

Description	A class representing user information.	

Syntax	<pre>[DataContract]
	class UserInfo : BaseInfo
	{
	int Key
	string UserID
	string FirstName
	string LastName
	string PINCode
	DateTime ValidFrom
	DateTime ValidUntil
	AccessFlags AccessFlags
	}</pre>	

Properties	int Key	The user database key.
	string UserID	The user ID.
	string FirstName	The user's first name.
	string LastName	The user's last name.
	string PINCode	The user's PIN code.
	DateTime ValidFrom	The date and time that the user is valid from.
	DateTime ValidUnitl	The date and time that the user is valid until.
	AccessFlags AccessFlags	The user access flags.

Appendix - Enabling HTTPS

To enable HTTPS (secure HTTP communication using TLS/SSL), a valid certificate must be installed on the server. A self-signed certificate can be generated with MakeCert if no valid certificate is available. MakeCert is available as part of the Windows 8 SDK.

How to

> **Tip:** If a certificate is already installed on the server, go to Step 8.

1. Run the command prompt as administrator.
2. Go to the folder where `MakeCert.exe` exists.

   For 64-bit computers:

   ```
   cd C:\Program Files (x86)\Windows Kits\8.0\bin\x64
   ```

   For 32-bit computers:

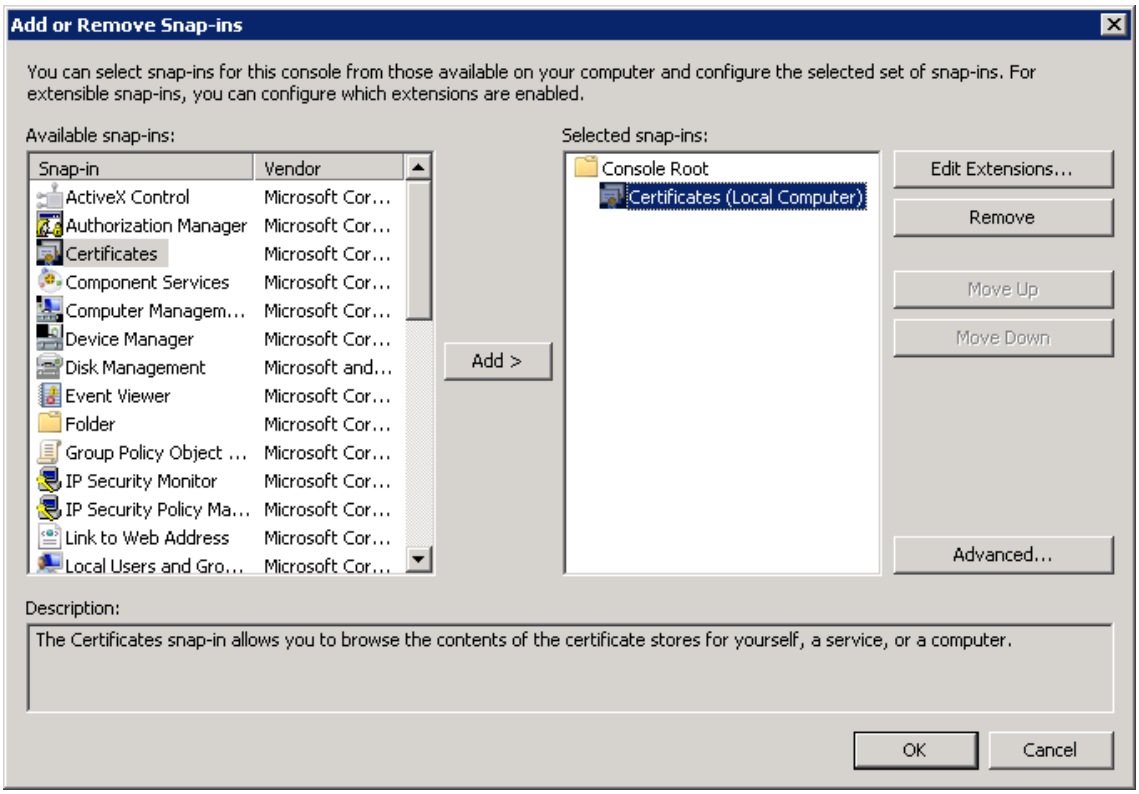
   ```
   cd C:\Program Files (x86)\Windows Kits\8.0\bin\x86
   ```

3. Run the command:

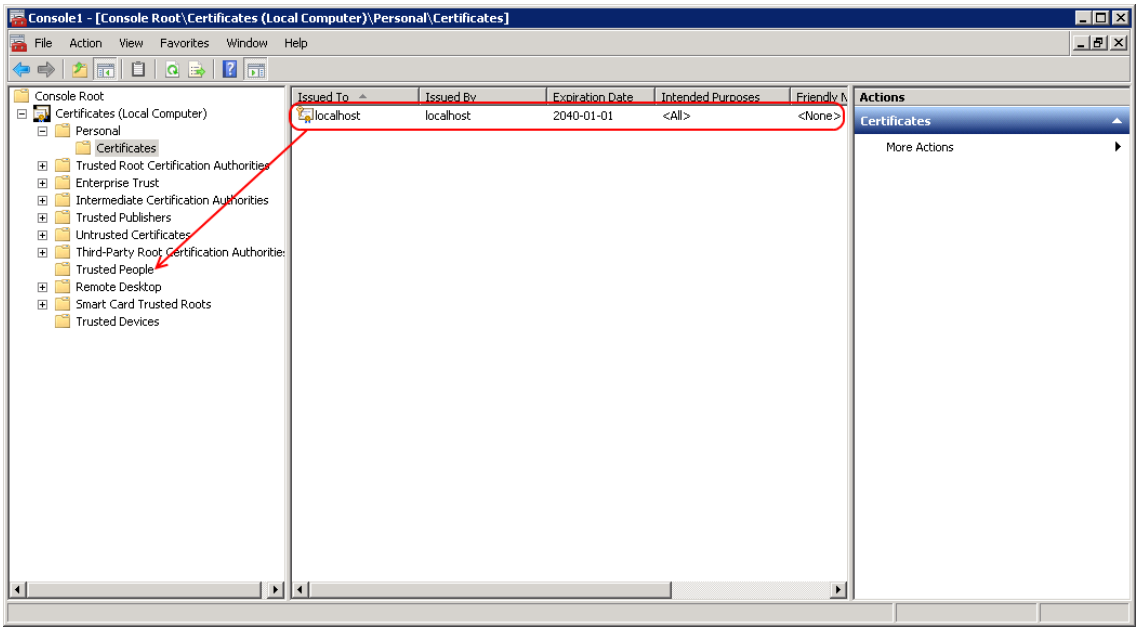
   ```
   makecert -sk MyCert -ss MY -sr LocalMachine -n CN=localhost -sky exchange -r
   ```

   to create a self-signed certificate.
4. Run `mmc`.

   MMC is the Microsoft Management Console.
5. In the MMC, select **File > Add/Remove Snap-in**.
6. Select to add the **Certificates**.



7. When prompted, select **Computer account** then **Local computer**.
8. Click **Finish** then **OK**.
9. Copy and paste your certificate from **Personal** to **Trusted People**.



10. Run the following in a command prompt as administrator:

```
netsh http add sslcert ipport=0.0.0.0:P certhash=C appid={G}
```

where:

P	The port where you wish to install SSL. The zeros stand for the local machine and P is the port. This should be the port where the Unison HTTPS Access Service is running. The default port is 8000 but this is configurable in the Unison Access Service driver.
C	The thumbprint of your certificate. To get the thumbprint of your certificate, open it and select the **Details** tab. Make sure to remove any spaces.
G	A unique GUID to identify the owning application. A GUID can be obtained at http://www.randomguid.com/