



Contents

<i>Introduction</i>	1
<i>Import prerequisites</i>	1
<i>Tables overview</i>	1
<i>Tables defined</i>	2
<i>Triggers</i>	9
<i>Data manipulation</i>	11
<i>Database connectivity</i>	15
<i>Troubleshooting</i>	17
<i>Printing a report on records downloaded to controllers</i>	17
<i>Trademark</i>	17

Introduction

This document is intended to be a guide to importing and linking external database systems to personnel data in the Facility Commander Wnx database. Throughout this document, reference to Facility Commander Wnx is represented as “FCWnx” in text content to avoid repetition.

Data must be extracted from a source, formatted to conform with the appropriate FCWnx record structure, and inserted into the SecurePerfect database. For example, if personnel information (such as name, address, department, identification number, and employment status) is already available in a Human Resources database, that information can be imported into the SecurePerfect database.

When importing data into the SecurePerfect database, a download to the controller occurs only in the following scenarios:

- data is inserted into the `BadgeTable`, `BadgeUserField`, `PersonAccessRightMapTable`, or `PersonTable`
- data is deleted from the `BadgeTable`, `BadgeUserField`, or `PersonAccessRightMapTable`
- an update occurs in the `BadgeTable`, `BadgeUserField`, `BadgeAccessTable` or `PersonTable`

If data is inserted, updated, or deleted from an appropriate table, a database trigger fires and calls an extended stored procedure. The controller must be online to successfully complete a download to the controller database.

Note: If the controller is offline, FCWnx routes the import data to the `OfflineDownload` table. When the controller returns to online, the information is downloaded to the controller database.

Six extended stored procedures are used:

1. `xp_BadgeDelete`
2. `xp_BadgeInsert`
3. `xp_BadgeUpdate`
4. `xp_PersonAccessRightsMapDelete`
5. `xp_PersonAccessRightsMapInsert`
6. `xp_PersonTableUpdate`

The extended stored procedures are located in the master database of the SPSQL instance on the SQL 2000/2005 Server computer. These extended stored procedures are bound to `xspDatabaseDownload.dll` located in the

`\Program Files\GE\FCWnx`

folder or your installed path. The `xspDatabaseDownload.dll` is registered to the Microsoft SQL 2000/2005 services. If you need to uninstall FCWnx, you must shut down SQL 2000/2005 services to release the `xspDatabaseDownload.dll` from use.

Note: FCWnx v7.0 does not support ACU controllers.

Import prerequisites

To perform the required import/export procedures, the following knowledge base is essential:

- Structured Query Language (SQL) and SQL 2000/2005 Server
- Programming skills in Visual Basic, C++, or other program that can import data into an SQL 2000/2005 database
- Access control concepts

Tables overview

The database tables involved in importing, exporting, or linking processes are as follows:

`PersonTable`

This table stores the minimum data needed to represent a person in the SecurePerfect database.

`UserFieldTable`

This table stores user-definable characteristics for a specific person in the `PersonTable`.

BadgeTable

This table stores data representing a badge (credential) that has been inserted into the system.

Note: A credential can exist without being assigned to a person.

BadgeUserField

This table stores user-definable characteristics for a specific badge (credential) in the BadgeTable.

DefinedBadgeStatus

This table stores user-definable badge (credential) status.

BadgeAccessTable

This table stores data representing a badge's (credential's) last valid access that has been inserted into the system.

PersonAccessRightMapTable

This table maps the person to a specific access right in the AccessRightTable.

FacilityTable

This table stores data representing facility identification.

DepartmentTable

This table stores data representing department and facility identification.

PersonTypeTable

This table is an association table; it stores personnel-type information such as Permanent or Contractor. Nothing is imported into it.

CompanyCode

This table stores company codes for ACU controller credentials.

Note: Refer to the SQL 2000 Enterprise Manager or the SQL 2005 Management Studio for detailed information about the FCWnx database schema.

Tables defined

PersonTable

The PersonTable Triggers are listed in [Table 14, PersonTable Triggers](#) on page 10. A PersonTable record is downloaded to the controller's database if there is an associated badge (credential) and an access right. If a PersonTable record has a credential but no access rights, a download to the controller does NOT occur. If a PersonTable record has an access right but no credential, a download to the controller does NOT occur.

When importing personnel data into the SecurePerfect database, start with the PersonTable. The PersonTable holds common identifying characteristics for individuals who receive or have received credentials.

[Table 1](#) lists all column names in the PersonTable along with the type, default value, and description.

See [page 10](#) for the triggers associated with the PersonTable.

Table 1. PersonTable

Column Name	Type	Nullable	Default	Description
Id (auto identity)	int	Not Null		Primary key
FirstName	nvarchar(64)	Null		Person's first name
LastName	nvarchar(64)	Not Null		Person's last name
MiddleName1	nvarchar(32)	Null		Person's first middle name
MiddleName2	nvarchar(32)	Null		Person's second middle name
Initials	nvarchar(3)	Null		First character from firstname, middlename, and middlename2
EmployeeNumber	nvarchar(12)	Not Null		Information that uniquely identifies person (Number must be at least four characters.)
Address1	nvarchar(64)	Null		Location of person or e-mail
Address2	nvarchar(64)	Null		Location of person or e-mail
Address3	nvarchar(64)	Null		Location of person or e-mail
Address4	nvarchar(64)	Null		Location of person or e-mail
Address5	nvarchar(64)	Null		Location of person or e-mail

Table 1. PersonTable (continued)

Column Name	Type	Nullable	Default	Description
Telephone	nvarchar(14)	Null		Person's phone number
PersonTypeId	int	Not Null		Foreign key to PersonTypeTable
DepartmentId	int	Null		Department to which person belongs (See DepartmentTable for Id.)
Traced	tinyint	Not Null	0	Is the person being traced? 0 = no 1 = yes
Photo	nvarchar(255)	Null		Name of file containing photographic image of person; name format is <personId.jpg>
FacilityId	int	Not Null	1	Foreign key to FacilityTable
Modified ¹	float	Not Null	0	Date and time record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication
ExtendUnlockTime	tinyint	Not Null	0	For extended unlock time on the reader
TraceAlarmMessageId	int	Null		Reserved
EscortRequired	tinyint	Not Null	0	Reserved
PassiveAPB	tinyint	Not Null	0	Allows access regardless of APB status
AccessOnAPBviolation	tinyint	Not Null	0	Allows access with APB violation
ConditionalUnlock	tinyint	Not Null	0	Allows access even if facility locked due to mode change or event that caused a normal schedule override. This person must also be assigned a credential that has the FollowConditionalUnlock field selected. (See BadgeTable for FollowUnconditionalUnlock.)
Title	nvarchar(50)	Null		Reserved
SuspendAllBadges	tinyint	Not Null	0	Reserved

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you would get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

UserFieldTable

The UserFieldTable table stores user-definable characteristics for a specific person in the PersonTable. The PersonTable and the UserFieldTable have a one-to-one relationship. For every record in the PersonTable, there must be a corresponding record in the UserFieldTable. The minimum information for the UserFieldTable consists of the corresponding PersonTableId in the PersonId column of the PersonTable and the time the record was changed in the Modified column.

Table 2, UserFieldTable on page 3 lists all column names along with the type, default value, and description.

Table 2. UserFieldTable

Column Name	Type	Nullable	Default	Description
PersonId	int	Not Null		Foreign key to the PersonTable
Value1	nvarchar(64)	Null		User-definable data
:				
:				
Value 90	nvarchar(64)	Null		User-definable data
Modified ¹	float	Not Null	0	Date and time record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

BadgeTable

The `BadgeTable` holds information on a badge (credential) that has been inserted into the system.

[Table 3](#) lists all column names in the `BadgeTable` along with the type, default value, and description. Following the table is a list of columns with additional import information.

See [page 9](#) for the triggers associated with the `BadgeTable`.

Table 3. `BadgeTable`

Column Name	Type	Nullable	Default	Description
Id	int	Not Null		Primary key
Description	nvarchar(64)	Not Null		Unique descriptive text
PersonId	int	Null		Person record assigned to the credential record (This is not a foreign key. A credential can exist without being assigned to a person.)
EncodedNumber	nvarchar(20)	Not Null		Number encoded into the card. (Only numbers and blanks are allowed and all numbers must be contiguous with any blanks leading all numbers.)
AliasNumber	nvarchar(20)	Not Null		Number that may be used to hide encoded number
Status	tinyint	Not Null	1	Credential's status (one of the following): 1 = Active 2 = Issuable 3 = Suspended 4 = Lost 5 = Remake 6 = Guard Tour
PIN	nvarchar(8)	Null		Personal Identification Number (Only numbers and blanks are allowed and all numbers must be contiguous with any blanks leading all numbers.)
ReturnDate ¹	float	Null		Date credential was last unassigned from a person - Null when it was never assigned
IssueDate ¹	float	Null	0	Date the credential was assigned or the date the credential becomes active ²
DueDate ¹	float	Null	0	Date that credential is no longer allowed access
FacilityId	int	Not Null	1	Foreign key to the <code>FacilityTable</code>
FollowExtendUnlock	tinyint	Not Null	0	Causes the reader to follow the Extended Unlock Time
Modified ¹	float	Not Null	0	Date and time when record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication
CompanyCodeId	int	Null		(ACU controller only) Company/Facility code
CardNumber	nvarchar(20)	Null		(ACU controller only) Number encoded into the card. (Only numbers and blanks are allowed and all numbers must be contiguous with any blanks leading all numbers.)
CardIssueNumber	nvarchar(2)	Null		(ACU controller only) Issue number of the card
BadgeStatusId	int	Not Null	1	Id of the record in the <code>DefinedBadgeStatus</code> table
FollowConditionalUnlock	tinyint	Not Null	0	Allows access even if facility locked due to mode change or event that caused a normal schedule override. This credential must also be assigned to a person that has the <code>ConditionalUnlock</code> field selected. (See <code>PersonTable</code> for <code>UnconditionalUnlock</code> .)

1. The `Modified` column and all columns representing a date are decimal numbers with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use `getdate()` or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using `'select cast(37328.332404243825 as datetime) - 2'`.
2. A credential set to Active before the issue date results in a credential that provides valid access.

Importing data into the `BadgeTable` to create a new credential without the use of the FCWnx software can be accomplished. The sections that follow provide additional information required to successfully import data into the `BadgeTable`.

EncodedNumber

Number stored in the credential and used by readers to identify a specific credential. The encoded number is built into the card. When a card is issued, the number is read from the card and inserted into the `BadgeTable`. The encoded number type is `nvarchar` and must be between 4 and 20 contiguous numeric characters in length. If you plan to import encoded numbers, they must be unique and between 4 and 20 characters in length.

The `EncodedNumber` column cannot be updated. Once data is inserted into the column, it cannot be changed.

AliasNumber

Any artificially created number that is used to hide the encoded number, or a copy of the encoded number. If aliasing is turned off, both columns, `Encoded Number` and `AliasNumber`, have the same value. The alias number is the number displayed to operators of FCWnx. This allows FCWnx to 'hide' the real credential encoded numbers from anybody using FCWnx.

The `AliasNumber` column cannot be updated. Once data is inserted into the column, it cannot be changed. For more information on aliasing, see the FCWnx Online Help.

PIN

PIN is only used with a credential and keypad reader. The PIN must be four digits.

Status

Can have values from 1 to 6

where:

- 1 Indicates credential is active with a valid `PersonId` in the `PersonId` column.
- 2 Indicates credential is issuable and the `PersonId` column is null.
- 3 Indicates credential is suspended. A suspended credential does not have access when presented to any reader.
- 4 Indicates credential is lost. A lost credential does not have access on any reader.
- 6 Indicates that this is a Guard Tour credential.

When creating a credential for the first time, you do not have to assign it to a person in the `PersonTable` (`PersonId` column is nullable). If the `PersonId` column is left null, the row is not downloaded to the controller.

All columns representing a date are decimal numbers with a zero(0) date equivalent to the date 1899-12-30 00:00:00.0000. SQL server uses a zero(0) equivalent date of 1900-01-01 00:00:00.000. If you use `getdate()` or any other date function in SQL server, you must subtract 2 to make the time consistent with other tables in the database.

ReturnDate

Date the credential is unassigned from a person. The `ReturnDate` value is of type float and a decimal representation of a date.

The `ReturnDate` column is informational only which means that changing this column does not cause a download to the controller to occur. The status can be set to 2 to make the credential active and issuable. The status can be set to 3 (suspended) to remove credential access. Once the status is updated, the controller database is updated.

IssueDate

Date the credential is assigned or the date the credential becomes active. The `IssueDate` column is of type float and a decimal representation of a date. When the `IssueDate` and the current date are the same, the controller database automatically allows the credential to have access.

DueDate

Date a credential is no longer allowed access. The `DueDate` column is of type float and a decimal representation of a date. When the `DueDate` date and the current date are the same, the controller's database automatically invalidates that credential's access. The maximum `DueDate` cannot exceed the current date plus 9 years. The status column does not automatically change when the `DueDate` is greater than or equal to the current date.

BadgeAccessTable

The `BadgeAccessTable` stores data indicating location and date/time of last valid access. The `BadgeAccessTable` and the `BadgeTable` are closely linked. For each record in the `BadgeTable` a record in the `BadgeAccessTable` must exist.

[Table 4](#) lists all column names in the `BadgeAccessTable` along with the type, default value, and description.

Table 4. `BadgeAccessTable`

Column Name	Type	Nullable	Default	Description
<code>BadgeId</code>	int	Not Null	0	Primary key
<code>LastAccess¹</code>	float	Null	0	Date and time when a valid badge (credential) accessed reader

Table 4. BadgeAccessTable (continued)

Column Name	Type	Nullable	Default	Description
APBStatus	int	Not Null	0	Current global anti-passback status: 0 = Neutral 1 = In 2 = Out
TASStatus	int	Not Null	0	Current global time and attendance status: 0 = Neutral 1 = In 2 = Out
ReaderId	int	Null	0	Reader that was last accessed
CurrentRegionId ²	int	Null	1	Credential accessed a reader in the current Region
Modified ¹	float	Not Null	0	Date and time when record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication
SecureAreaId	int	Null		(ACU controller only) Keeps track of the area in which the credentialholder is located

1. The Modified column and all columns representing a date are decimal numbers with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.
2. This column pertains to FCWnx Global Edition and stores Region data.

During an import, the following values are set:

BadgeId

Corresponding BadgeId in the BadgeTable.

Last Access

Is set to zero(0) by default. The zero represents a credential that has never accessed a reader.

APBStatus

TASStatus

Is set to zero(0) which is neutral. For more information on Time and Attendance, refer to the FCWnx Online Help.

ReaderId

Is set to null when importing in the initial record.

CurrentRegionId

Leave as default value.

Note: After the initial record has been imported, do not manipulate the BadgeAccessTable directly.

BadgeUserField

The BadgeUserField stores user-definable characteristics for a specific person in the BadgeTable. The BadgeTable and the BadgeUserField have a one-to-one relationship. For every record in the BadgeTable, there must be a corresponding record in the BadgeUserField. The minimum information for the BadgeUserField consists of the corresponding BadgeTableId in the BadgeId column of the BadgeTable and the time the record changed in the Modified column.

Table 5 lists all column names in the BadgeUserField along with the type, default value, and description.

Table 5. BadgeUserField

Column Name	Type	Nullable	Default	Description
BadgeId	int	Not Null		Foreign key to the BadgeTable
BadgeUserField1	nvarchar(64)	Null		User-definable data
:				
:				
BadgeUserField20	nvarchar(64)	Null		User-definable data
Modified ¹	float	Not Null	0	Date and time record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

DefinedBadgeAccess

The DefinedBadgeAccess maps user-defined badge (credential) status to either active or suspended.

The minimum information for the DefinedBadgeAccess consists of the corresponding BadgeStatusId in the BadgeId column of the BadgeTable and the time the record changed in the Modified column.

[Table 6, DefinedBadgeAccess Table](#) on page 7 lists all column names in the DefinedBadgeAccess along with the type, default value, and description.

Table 6. DefinedBadgeAccess Table

Column Name	Type	Nullable	Default	Description
Id	int	Not Null		Foreign key to the BadgeTable (BadgeStatusId column)
BadgeStatusEnum	tinyint	Null		Next available status number
Description	nvarchar(64)	Null		Unique descriptive text
BadgeLineColor	tinyint	Not Null	0	Reserved
Modified ¹	float	Not Null	0	Date and time record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

PersonAccessRightMapTable

The PersonAccessRightMapTable maps the person to a specific access right in the AccessRightTable.

[Table 7, PersonAccessRightMapTable](#) on page 7 lists all column names in the PersonAccessRightMapTable along with the type, default value, and description.

See [page 10](#) for the triggers associated with the PersonAccessRightMapTable.

Table 7. PersonAccessRightMapTable

Column Name	Type	Nullable	Default	Description
PersonId	int	Not Null		Foreign key to the PersonTable
AccessRightId	int	Not Null		Foreign key to the AccessRightTable
Modified ¹	float	Not Null	0	Date and time the record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getdate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

When inserting or updating into the PersonAccessRightMapTable, make sure that a PersonId from the PersonTable and an AccessRightId from the AccessRightTable exist. Access is granted to a person record, not a badge (credential); keep this in mind when you are working with the SecurePerfect database.

A Person record can be associated with multiple credentials; however, a credential CANNOT have multiple person records. The PersonAccessRightMapTable is the table that grants an individual person access to specific areas already set up by FCWnx software.

You must create Access Rights within the FCWnx system. Do not try to create and import new access rights.

FacilityTable

The `FacilityTable` stores data representing facility identification.

[Table 8](#) lists all column names in the `FacilityTable` along with the type, default value, and description.

Table 8. `FacilityTable`

Column Name	Type	Nullable	Default	Description
Id (auto identity)	int	Not Null		Primary key
Description	nvarchar(64)	Not Null		Unique descriptive text
Modified ¹	float	Not Null	0	Date and time the record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use `getdate()` or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using `'select cast(37328.332404243825 as datetime) - 2'`.

DepartmentTable

The `DepartmentTable` stores data representing department and facility identification.

[Table 9](#) lists all column names in the `DepartmentTable` along with the type, default value, and description.

Table 9. `DepartmentTable`

Column Name	Type	Nullable	Default	Description
Id (auto identity)	int	Not Null		Primary key
Description	nvarchar(64)	Not Null		Unique descriptive text
FacilityId	int	Not Null	1	Foreign key to the <code>FacilityTable</code>
Modified ¹	float	Not Null	0	Date and time record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use `getdate()` or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using `'select cast(37328.332404243825 as datetime) - 2'`.

PersonTypeTable

The `PersonTypeTable` stores data representing personnel type such as `Permanent` or `Temporary`.

[Table 10, `PersonTypeTable`](#) on page 8 lists all column names in the `PersonTypeTable` along with the type, default value, and description.

Table 10. `PersonTypeTable`

Column Name	Type	Nullable	Default	Description
Id (auto identity)	int	Not Null		Primary key
Description	nvarchar(64)	Not Null		Unique descriptive text
BadgeDesignId	int	Null		Badge (credential) design used by default for printing credentials for this person type
FacilityId	int	Not Null	1	Foreign key to the <code>FacilityTable</code>
Modified ¹	float	Not Null	0	Date and time the record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use `getdate()` or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using `'select cast(37328.332404243825 as datetime) - 2'`.

CompanyCode

The CompanyCode table stores data representing the company codes for ACU controller credentials.

[Table 11, CompanyCode](#) on page 9 lists all column names in the CompanyCode table.

Table 11. CompanyCode

Column Name	Type	Nullable	Default	Description
Id (auto identity)	int	Not Null		Primary key
CompanyCode	nvarchar(10)	Null	1	Number representing the company code for ACU controller credentials
Modified ¹	float	Not Null	0	Date and time the record was last modified
Rowguid	uniqueidentifier	Not Null		Created by SQL for replication

1. The Modified column is a decimal number with a zero(0) date equivalent to the date 1899-12-30 00:00:00.000. SQL Server uses a zero(0) date equivalent date of 1900-01-01 00:00:00.000. If you use getDate() or any other date function in SQL Server, you must subtract 2 to make the time consistent with other tables in the database. If the modified value retrieved from a table is 37328.332404243825, you get the correct date by using 'select cast(37328.332404243825 as datetime) - 2'.

Triggers

A trigger is called to download changes to a controller whenever fields in certain columns are deleted, inserted, or updated. Only three tables contain triggers that deal with the import process: BadgeTable, PersonAccessRightMapTable, and PersonTable.

BadgeTable triggers

Table 12. BadgeTable Triggers

Action	Trigger Column(s)	Trigger Fired	Extended Stored Procedure
Delete	All columns in the BadgeTable	GETR_Badge_Delete	xp_BadgeDelete
Insert	Minimum of: EncodedNumber, AliasNumber, Status, Modified, FacilityId, IssueDate If ACU controller must also have CompanyCodeId, CardNumber and CardIssueNumber	GETR_Badge_Insert	xp_BadgeInsert
Update	Any one of the following: Status, DueDate, PIN, PersonId, IssueDate, FollowExtendUnlock, CardIssueNumber	GE_TRIG_Badge_Update	xp_BadgeUpdate

The BadgeTable Triggers are listed in [Table 12](#).

GETR_Badge_Delete

A delete trigger on the BadgeTable. When the GETR_Badge_Delete trigger fires, it calls the xp_BadgeDelete extended stored procedure that calls the xspDatabaseDownload.dll to remove the badge (credential) from the controller's database.

When a record is deleted from the BadgeTable, the corresponding record is removed from the BadgeAccessTable.

GETR_Badge_Insert

If this record is for an ACU controller, then CompanyCodeId, CardNumber, and CardIssueNumber must have an entry that includes the EncodedNumber which is the concatenation of the CompanyCodeId from the CompanyCode table and the CardNumber.

An insert trigger on the BadgeTable. When the GETR_Badge_Insert trigger fires, it checks if the FacilityId exists. If the FacilityId exists, the transaction proceeds normally. If the FacilityId does not exist, a one(1) is inserted for the FacilityId. A one(1) for FacilityId means Ignore Facilities. After the facility check, the xp_BadgeInsert extended stored procedure is called. If there is a PersonId and the person record has access rights, the badge (credential) information is downloaded to the controller. If the PersonId field is null and/or the person record has no access rights, a download to the controller does not occur.

When a record is inserted into the BadgeTable, you must insert a corresponding record into the BadgeAccessTable. The record is the BadgeId of the record inserted into the BadgeTable, LastAccess with a value of zero(0), APBStatus with a value of zero(0), TASStatus with a value of (0), a ReaderId set to zero(0), and a CurrentRegionId with a default value of one(1).

GE_TRIG_Badge_Update

If this record is for an ACU controller, then CompanyCodeId, CardNumber, and CardIssueNumber must have an entry that includes the EncodedNumber which is the concatenation of the CompanyCodeId from the CompanyCode table and the CardNumber.

An update trigger on the BadgeTable. When the GE_TRIG_Badge_Update trigger fires, it checks the EncodedNumber or AliasNumber, Status, DueDate, PIN, IssueDate, and PersonId. If the EncodedNumber or AliasNumber is updated, an error is fired and the transaction is rolled back. If the EncodedNumber and AliasNumber are untouched, then the trigger checks if the Status, IssueDate, DueDate, PIN, or PersonId columns have been updated. If one or more of the previously noted columns have been updated, a download to the controller occurs. The download to the controller occurs only if the badge (credential) is associated with a valid personId and the person has access rights.

PersonAccessRightMapTable triggers

Table 13. PersonAccessRightMapTable Triggers

Action	Trigger Column(s)	Trigger Fired	Extended Stored Procedure
Delete	All	GETR_person_access_rights_map_delete	xp_PersonAccessRightsMapDelete
Insert	All	GETR_person_access_rights_map_insert	xp_PersonAccessRightsMapInsert
Update	Not Applicable		

The PersonAccessRightMapTable Triggers are listed in [Table 13](#).

GETR_person_access_rights_map_delete

A delete trigger on the PersonAccessRightMapTable. When this trigger fires, it calls the xp_PersonAccessRightsMapDelete extended stored procedure and the associated badge (credential) record is removed from the controllers.

GETR_person_access_rights_map_insert

An insert trigger on the PersonAccessRightMapTable. When this trigger fires, the xp_PersonAccessRightsMapInsert extended stored procedure inserts a badge (credential) record into the controllers.

PersonTable triggers

Table 14. PersonTable Triggers

Action	Trigger Column(s)	Trigger Fired	Extended Stored Procedure
Delete	Not Applicable		
Insert	FacilityId	GETR_Facility_PersonTable	none
Update	Traced or ExtendUnlockTime	GETR_UpdatePersonStatus	xp_PersonTableUpdate

Note: The Triggers are detailed in the sections that follow.

The PersonTable Triggers are listed in [Table 14](#).

GETR_Facility_PersonTable

An update and insert trigger on the PersonTable. The trigger checks if the updated or inserted FacilityId exists. If the FacilityId does not exist, the FacilityId is set to 1 (ignore facilities). If the FacilityId does exist, then the transaction proceeds normally.

GETR_UpdatePersonStatus

An update trigger on the PersonTable Traced column or the PersonTable ExtendUnlockTime column. The trigger checks if the Traced column has changed during the update. If the column has been updated, then the xp_PersonTableUpdate extended stored procedure is called and a download to the controller occurs.

Data manipulation

There are five possible scenarios where data is imported or updated into the `SecurePerfect` database:

1. A one-time mass import of people and badges (credentials)
2. A one-time mass update of people and credentials
3. Inserting individual records on a continuous basis from an application outside of FCWnx
4. Updating individual records on a continuous basis from an application outside of FCWnx
5. Deleting records from the database

The controllers must be online for any insert, update, or delete to complete successfully. If the controller is offline, FCWnx routes the import data to the `OfflineDownload` table. When the controller returns to online, the information is downloaded to the controller database.

Import methods

Data can be imported into the database using SQL 2000/2005 bulk insert commands or DTS. If bulk insert or DTS is used, they must be configured to allow triggers to be fired. The setting for the bulk insert is `FIRE_TRIGGER`. For DTS, the `FastLoad` option must be set to false.

Note: If you plan to import data into the `BadgeTable`, `PersonAccessRightMapTable`, or `PersonTable`, you must use a method to insert or update records that allows the triggers to be fired.

There are other methods to connect using Visual Basic or Visual C++.

Sources of additional information:

- SQL Books Online
- The following topics of Microsoft documentation: **ADO**, **OLEDB**, and **connect**.

If you choose to install the samples during installation of SQL Server 2000, the following is a helpful sample:

```
<PATH>\Microsoft SQL Server\80\Tools\DevTools\Samples\ado\vb\intro
```

Mass import of people and badges (credentials)

Mass imports and updates should be executed only during off-peak hours.

To mass import people and credentials:

1. Insert records into the `PersonTable`. If your person records contain `DepartmentId` or `FacilityId`, make sure that each ID exists in the `DepartmentTable` and `FacilityTable` respectively.
2. For every record inserted into the `PersonTable`, a corresponding record needs to be added to the `UserFieldTable`. The minimum columns needed are the `PersonId` and the `Modified` columns. The `Modified` column is the date and time the record was last modified.
3. Insert the credential records into the `BadgeTable` and `BadgeUserField`. If the insert is missing a `PersonId`, there is no download to the controller. If aliasing is turned on, the alias number and the encoded number are different. If aliasing is turned off, the alias number and the encoded number are the same.
4. Insert a corresponding `BadgeAccessTable` record. Each record in the `BadgeTable` must have a corresponding record in the `BadgeAccessTable`.
5. Grant the Person access rights by inserting records into the `PersonAccessRightMapTable`. The `AccessRightId` must exist in the `AccessRightTable`. Access rights are created in FCWnx only, do not try to create access rights outside of FCWnx.

Refer to [Figure 1](#) and [Figure 2](#) for examples of importing.

Figure 1. Flowchart - importing person records only

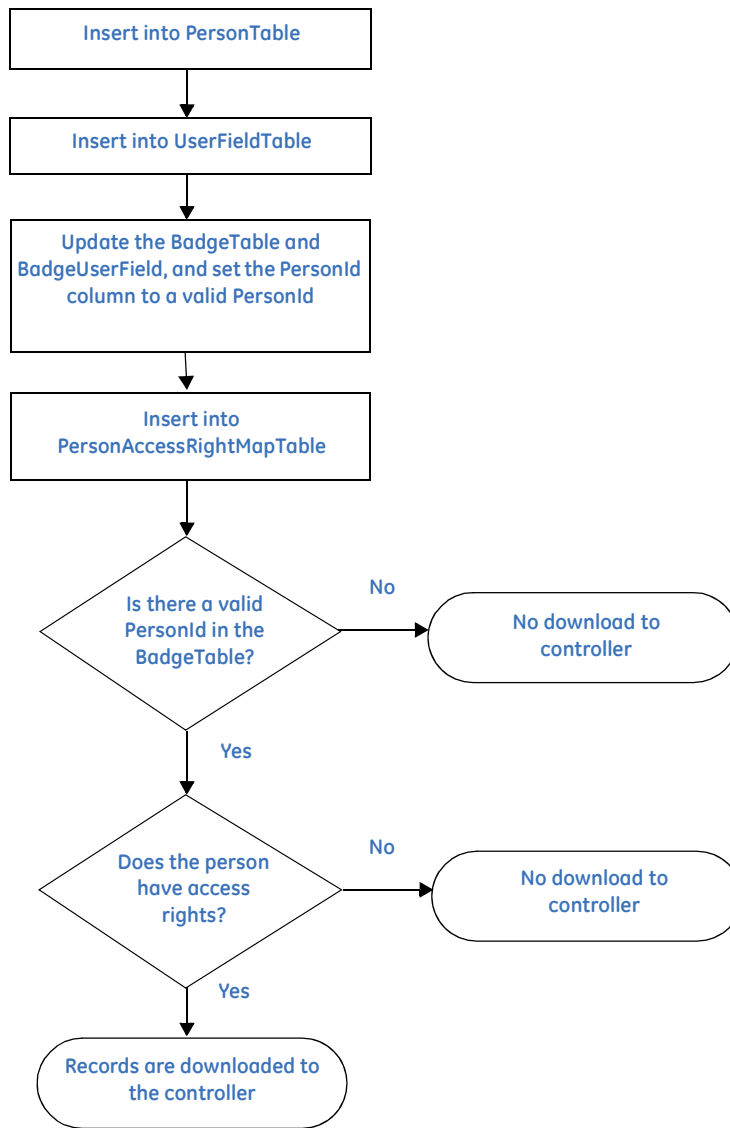
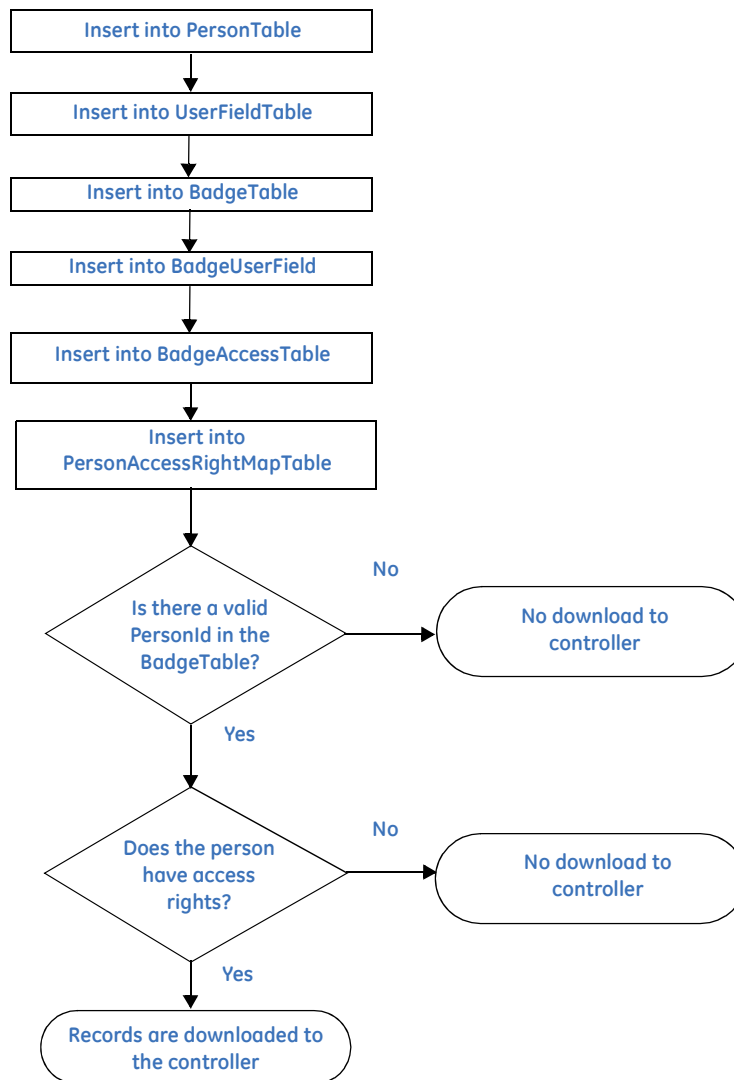


Figure 2. Flowchart - importing person records and badge records



Mass update of people and badges (credentials)

To mass update people and credential records:

1. Update the records in the `PersonTable`. If you update the `DepartmentId` or `FacilityId`, make sure that the ID exists in the `DepartmentTable` and the `FacilityTable`, respectively.
2. Update the `UserFieldTable` if needed or desired. To update a `UserField` record, you need the corresponding `PersonID` from the `PersonTable`.
3. Update the credential records in the `BadgeTable` and `BadgeUserField`, if needed. If the updated record is missing a `PersonId` or an access right, there is no download to the controller. The encoded number and the alias number cannot be updated.
4. Update the Person's access rights by inserting records into the `PersonAccessRightMapTable`. The `AccessRightId` must exist in the `AccessRightTable`.

Note: Do not update the `BadgeAccessTable`.

Access rights are created only within FCWnx.

Note: Do not try to create access rights outside of the FCWnx system.

Continuous insert or update of records

A continuous insert or update of records from a system outside of FCWnx can be accomplished by writing a custom program to read the data from the outside application, connect to the `SecurePerfect` database, and insert or update the data into the appropriate table(s). It is recommended that the outside application and the database have some sort of commonality. For example, the employee number of the `PersonTable` in the `SecurePerfect` database matches up to some identification number from the outside application. If the outside application is a Human Resources system and Person X has been assigned the identification number 234567, then 234567 may be inserted into the `PersonTable` as the employee number.

Continuous insert

To insert individual records on a continuous basis:

1. Decide how the record in the `SecurePerfect` database is to link to the record outside of the `SecurePerfect` database.
2. Write a program to connect to the `SecurePerfect` database and import the data. The order of operations for an import is:
 - a. Insert the record into the `PersonTable`.
 - b. Insert a corresponding record into the `UserFieldTable`.
 - c. Insert a record into the `BadgeTable` and corresponding record in the `BadgeAccessTable`.
 - d. Insert an access right into the `PersonAccessRightMapTable`, if applicable.

Continuous update

To update individual records on a continuous basis:

1. Decide how the record in the `SecurePerfect` database is to link to the record outside of the `SecurePerfect` database.
2. Write a program to connect to the `SecurePerfect` database and update the data.

The order of operations for an update varies but keep the following concepts in mind:

- a. The controller must be online for any insert, update, or delete to complete successfully.
- b. A download to the database occurs only if a badge (credential) record has a valid `PersonId` and the corresponding person record has access rights.

If the controller is offline, FCWnx routes the import data to the `OfflineDownload` table. When the controller returns to online, the information is downloaded to the controller database.

Deleting records from the database

Deleting a person

Deleting a person from the database can be accomplished by calling the `usp_delete_person` stored procedure from the `SecurePerfect` database and passing the appropriate `PersonId`. The `usp_delete_person` stored procedure deletes the badge (credential), access rights, as well as other records associated to the person.

Note: Do not attempt to delete a person directly from the table.

Example: `exec usp_delete_person 24` deletes the person with record ID 24 from the `PersonTable`.

Deleting a credential

Deleting a credential from the database can be accomplished by calling the `usp_delete_badge` stored procedure from the `SecurePerfect` database and passing the appropriate `BadgeId`. The `usp_delete_badge` stored procedure deletes the credential, access rights, as well as other records associated to the credential.

Note: Do not attempt to delete a credential directly from the table.

Example: `exec usp_delete_badge 264` deletes a credential record with record ID 264 from the `BadgeTable`.

Database connectivity

The following sections detail two methods of connectivity.

ODBC connection

If you use ODBC, you should have the SQL server administrator create a dedicated login and password, and assign the login to the appropriate rights and privileges. Your own dedicated connection to the database allows easier debugging of any application you build and more flexibility in development. **DO NOT USE** the FCWnx ODBC connection.

ODBC connections are configured using the **Control Panel** ODBC setup applet. ODBC connections can be made against any data source for which an ODBC driver has been installed. Visual C++ and Visual Basic ship with drivers for Text files, Access, FoxPro, Paradox, dBase, Excel, SQL Server, and Oracle. When you create an ODBC connection, it automatically receives a Data Source Name (DSN). The DSN is subsequently used to identify connections in data-source controls, such as ADO Data Control and RDO Remote Data Control.

To configure an ODBC data source for Windows 2000 Server:

1. Click **Start**, select **Settings, Control Panel, Administrative Tools**, then **Data Sources (ODBC)**.
2. Select the **User DSN** or **System DSN** tab. The **User DSN** tab lets you create user-specific Data Source Names and the **System DSN** tab lets you create data-sources available to all users.
3. Click **Add** to display a list of locally installed ODBC drivers.
4. Select the SQL server data source for your current driver.
5. Follow the instructions specific to the driver. After closing, the DSN is now available for use.
6. You should have a valid login and password. Make the login a user of the *SecurePerfect* database with appropriate rights and privileges to select, insert, update, and delete on the database. You can assign the new user the *db_owner* role (see SQL server books online) to give total access of the database to the new user.

ADO connection example

You can connect to the *SecurePerfect* database using Visual Basic and ADO.

The example in *Figure 3* uses the Microsoft ActiveX Data Objects. Add the ADO reference from Visual Basics Project toolbar, then select References and check the ADO reference. Add a text box and a command button.

There are other methods to connect using Visual Basic or Visual C++. Additional information can be obtained in the following topics of Microsoft documentation: ADO, OLEDB, and connect. If you choose to install the samples during installation of SQL Server 2000, the following is a helpful sample:

```
<PATH>\Microsoft SQL Server\80\Tools\DevTools\Samples\ado\vb\intro
```

Figure 3. Sample Program - ADO Connection

```

Option Explicit
Dim cn As New ADODB.Connection
Private Sub Command1_Click()
    On Error GoTo ErrHandler:
    Dim UserName As String
    Dim Password As String
    Dim ServerName As String
    Dim DBName As String

    UserName = "<any valid login>"
    Password = "<password for the valid login>"
    ServerName = "<server name>\<instance name>" 'YourServerName\SPSQL
    DBName = "SecurePerfect"

    'Set connection properties.
    cn.ConnectionTimeout = 25 'Set the time out.
    cn.Provider = "sqloledb" 'Specify the OLE DB provider
    cn.Properties("Data Source").Value = ServerName 'Set SQLOLEDB connection properties.
    cn.Properties("Initial Catalog").Value = DBName 'Set SQLOLEDB connection properties.
    cn.Properties("Integrated Security").Value = "SSPI" 'Set SQLOLEDB connection
    properties.

    'Change mousepointer while trying to open database.
    Screen.MousePointer = vbHourglass

    'Open the database.
    cn.Open
    Screen.MousePointer = vbDefault

    Text1.Text="Connected"
Exit Sub
ErrHandler:
    'Change mousepointer back to the default after open.
    Screen.MousePointer = vbDefault
    'Display the error message.
    MsgBox Err.Description, "Error "
    'End the program.

```

Troubleshooting

Use the DBTrigger diagnostic object to troubleshoot the external import, update, or delete process.

Turning on the DBtrigger diagnostic

Before you can capture and view diagnostic information, the appropriate diagnostic component must be turned on.

To turn on the DBtrigger diagnostic:

1. In the FCWnx system from the Application Group pane, select **Diagnostics**, then **Diagnostic Setting**.
2. Click **Search** in the toolbar to display a list of components that you can monitor.
3. Select the **DBTrigger** diagnostic on the Server computer.
Note: All diagnostic objects are prefixed with a machine name.
4. Select **Enable debug messages** check box and click **Save**.
5. When you are finished troubleshooting the system, don't forget to go back and **DISABLE** debug messages.

Viewing the diagnostics log

For each client, there is a default logfile (others can be created) for each day of the week such as `SPEEFriday spl`.

DiagView operates in "real time." That is, every time FCWnx writes an entry to the log file, DiagView automatically displays the latest message. By default, DiagView displays only the latest 1000 messages. The number displayed can be changed on the Preferences Form. All log files should be saved in the logs folder; it is easier to locate for backups and upgrades. It is a shared folder which means other clients can gain access to the log files.

To view the diagnostics log:

1. In the FCWnx system from the Application Group pane, select **Diagnostic**, then **Diagnostic Viewer**.
2. Select **Open** and then click **Ok** to open the current day's logfile.
3. When examining the log, you see messages similar to:
`BadgeTable record update`
This means the database trigger has fired and has successfully called the `xspDatabaseDownload.dll`.

Printing a report on records downloaded to controllers

You can print a report that lists all records (imported, updated, or deleted) that were uploaded to controllers.

To print a report listing all records downloaded to controllers:

1. In the FCWnx system from the main menu, select the **Reports** menu, then **Operator History**.
2. In the **Filter** field, enter the login name `Trigger`. The resulting report lists all records that were passed to the `xspDatabaseDownload.dll`.

Trademark

GE and the GE monogram are registered trademarks of General Electric.
Facility Commander Wnx product and logo are trademarks of GE Security.

Other trade names used in this document may be trademarks or registered trademarks of the manufacturers or vendors of the respective products.

Technical support

Toll-free: 888.GESECURITY (888.437.3287 in the US, including Alaska and Hawaii; Puerto Rico; Canada).
Outside the toll-free area: Contact your local dealer.

www.gesecurity.com