



# Security Commander API Installation and Operation Manual

<b>Copyright</b>	© 2015 UTC Fire & Security. All rights reserved.
<b>Trademarks and patents</b>	<p>The Security Commander API name and logo are trademarks of UTC Fire &amp; Security.</p> <p>Other trade names used in this document may be trademarks or registered trademarks of the manufacturers or vendors of the respective products.</p>
<b>Manufacturer</b>	<p>UTC Fire &amp; Security Australia Pty Ltd t/a Interlogix A UTC Building &amp; Industrial Systems company Ground Floor, 10 Ferntree Place, Notting Hill, VIC, 3168, Australia</p>
<b>Contact information</b>	For contact information, see <a href="http://www.interlogix.com.au">www.interlogix.com.au</a> .

# Content

<b>Important information.....</b>	<b>ii</b>
Purpose.....	ii
Scope.....	ii
Who should read this manual.....	ii
Related documentation .....	iii
Typographical conventions .....	iii
<b>Security Commander API.....</b>	<b>1</b>
Overview .....	1
System diagram .....	1
System components .....	2
System requirements .....	3
<b>Installation .....</b>	<b>4</b>
Overview .....	4
Installation procedures .....	4
<b>Operation .....</b>	<b>11</b>
Overview .....	11
Person records.....	11
Audit facilities .....	12
Security Commander alarms.....	12
Troubleshooting .....	13
<b>Using XML Files .....</b>	<b>15</b>
Overview .....	15
Creating new person records .....	17
Updating Person records .....	19
Deleting persons or badges .....	21
File naming and handling .....	23

# Important information

Security Commander API (Application-Program Interface) provides the ability to use external applications such as a Human Resource Management System (HRMS) for personnel and badge management, and then import the required data into Security Commander via appropriately configured Extensible Markup Language (XML) files. Security Commander API may also be used to transfer person records in bulk into the Security Commander system.

## Purpose

This Installation and Operation Manual describes how to install Security Commander API and use it to perform common personnel and badge operations.

In particular, the personnel and badge operations are performed using the customer's external application, such as an HRMS. The new or revised personnel data is then used to create XML files for export to Security Commander API, which then updates the appropriate Security Commander databases.

The process by which the customer's data is used to create the XML files will vary depending on the type of external database, and is not described in this manual. It is the responsibility of the customer to ensure that the XML files are provided in a manner that can be used by Security Commander API.

## Scope

This manual describes how to set up and use Security Commander API to perform typical operations, errors that may occur, and how to diagnose and correct faults.

This manual contains details of using XML files to perform person and badge operations, but it does not describe how to create XML files from external data.

## Who should read this manual

This manual is intended for:

- System administrators responsible for transferring data from external applications into Security Commander
- System integrators and developers

The material in this manual has been prepared for persons responsible for, and familiar with, the security needs of the customer facility.

## Related documentation

For more information, refer to the following:

- *Security Commander Installation Manual*: Provides information for Integration Technicians to set up, install, and configure a Security Commander system.
- *Security Commander Administration Manual and Operating Guide*: Provides information for both the system administrator and the operator to program, configure, and use the Security Commander system.
- *Security Commander Online Help*: Provides reference information, such as screen and field descriptions, along with instructions for system administrator duties, such as configuring Challenger panels.
- *Security Commander CCTV Interface Guide*: This manual provides interface instructions for CCTV equipment.
- *Security Commander Photo ID User Guide*: Provides instructions for users of the optional Photo ID package.

## Typographical conventions

This manual uses certain notational and typographical conventions to make it easier for you to identify important information.

**Table 1: Notational and typographical conventions**

Item	Example
Command sequences	Where appropriate, command sequences are abbreviated with the ">" symbol. For example, the command "Click Start, and then click Run" is written as "Click Start > Run".
Command alternatives	Many commands can be executed in a variety of ways including menu bar, tool bar, shortcut keys, right click, or Double-click. In general, commands are described from their menu bar location only, even when alternatives exist.
Keys	Capitalized, for example "press Enter".
Keystrokes	Text that you type is indicated in Courier New font. For example, "Type dcomcnfg".
Expanding a "tree" view	The word "expand" is used to indicate that selections may be hidden. For example, the command "Click the + box next to Computers" is written as "Expand Computers".
Notes	Notes alert you to information that can save you time and effort.
Cautions	Cautions are displayed to advise the user that failure to take or avoid a specified action could result in loss of data.



# Security Commander API

## Overview

Security Commander API enables data (such as personnel data contained in external databases) to be imported into the Security Commander system via XML files (appropriately formatted text files with an .XML file extension).

There are two typical scenarios in which Security Commander API may be used with Security Commander:

- Ongoing: The customer's HRMS is used to maintain person and badge records, which are imported into Security Commander via Security Commander API. As well, other external applications such as visitor badge control may also need to export data into Security Commander via Security Commander API.
- One-off: Security Commander API may be used on a one-off basis to import bulk quantities of data into Security Commander, after which these records are maintained in the Security Commander databases via the Security Commander user interface.

There is no provision in Security Commander API to transfer data from Security Commander to an HRMS.

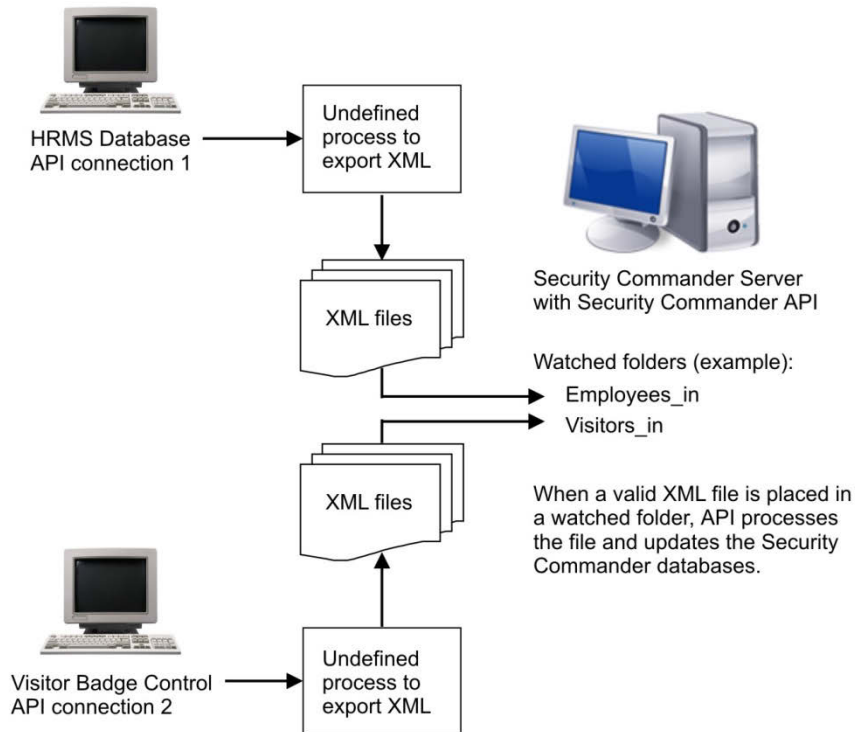
## System diagram

Figure 1 on page 2 depicts a sample system in which:

- Data is required to be imported into Security Commander from two sources, an HRMS and a visitor badge control system.
- Each data source is identified by a separate Security Commander API Connection record.
- The Security Commander API computer is shown as a Security Commander server, but it can be a client.
- Each data source has a separate watched folder on the Security Commander API computer. These watched folders must be protected against unauthorized use to unauthorized changes to person or badge data.
- The customer's IT staff or suitable applications create the XML files from the external data.
- An operator or application with authority to write data to the watched folders copies the XML files to their respective folders.
- Security Commander API processes the XML files and updates the Security Commander databases.

- Successfully processed XML files are automatically deleted (and not sent to the Windows Recycle Bin). Files that are not successfully processed are moved to the designated error folders and details are logged for fault diagnosis (XML files must not have a read-only attribute, or they can't be deleted or moved by Security Commander API).

**Figure 1: Data transfer from two API connections to Security Commander server or client computer via XML files**



## System components

The main components of Security Commander API are as follows:

- Alliance 8300 API Service: Service that runs in the background to enable the API connection.
- Security Commander API Connections form: Defines an Application Program Interface (API) connection record within Security Commander to control access to data. The API connections record identifies the login name, password, computer name, and operator that are allowed access.
- Alliance 8300 Directory Watcher: Service that runs in the background to monitor the user-defined 'watched' folder for the presence of an XML file.
- SPDirWatcher.log: Log file used for fault diagnosis.
- User-defined 'watched' folders: XML files placed in the watched folder are automatically processed. Write access to the watched folder must be controlled to prevent unauthorised use.
- User-defined 'error' folders: XML files that cannot be successfully processed are automatically moved here.



- SpDirWatcherConfigurator.exe: Application used to define the locations of Security Commander API files and folders and to identify the login name and password required for each API connection (there can be multiple API connections).
- XML files for data transfer: Created by the customer's IT staff or by suitable applications from external data using the guidelines contained in this manual (see "Using XML Files" on page 15).

## System requirements

Security Commander API runs on a Security Commander server or client computers, which are described in the *Security Commander Installation Manual*.

In addition to the listed requirements, Security Commander API requires:

- Microsoft .NET Framework (installation file is provided on the Security Commander CD).
- Security Commander version 02.02.00 or later.

# Installation

## Overview

Table 2 below describes the overall process of setting up a Security Commander server or client computer to use Security Commander API.

Unless otherwise noted, perform the tasks in the order they appear.

**Table 2: Main tasks to set up Security Commander API**

Number	Task	Reference
1.	Install Microsoft .NET Framework	"Installing Microsoft .NET Framework" below
2.	Install the API services	"Installing the API services" below
3.	Create or open an API Connection record in Security Commander	"Creating an API Connection in Security Commander" on page 5
4.	Configure SPDirWatcher	"Configuring SPDirWatcher" on page 6
5.	Start services	"Starting Alliance 8300 Directory Watcher service manually" on page 8

## Installation procedures

### Installing Microsoft .NET Framework

Microsoft .NET Framework is not installed as part of Security Commander, but may have been previously installed on the Security Commander API computer. Microsoft .NET Framework version 1.1 (or greater) is required to use Security Commander API.

#### To install Microsoft .NET Framework:

1. On the Security Commander computer that will host Security Commander API, check the Add/Remove Programs window to see if Microsoft .NET Framework 1.1 (or greater) is present. If a suitable version of Microsoft .NET Framework is not present, then you must install Microsoft .NET Framework 1.1.
2. If you need to install or upgrade Microsoft .NET Framework, navigate to the NET folder on the Security Commander CD, run the file DOTNETFX.EXE, and then follow the prompts.

### Installing the API services

#### To install the API services:

1. Navigate to the API folder on the Security Commander CD.
2. Double-click "Dir Watcher Setup.exe" to start installing the API services.
3. Click Next in the welcome screen.

4. Click Install in the Ready to Install window. Installation will start.
5. When completed, click Finish to complete installation.

### Creating an API Connection in Security Commander

Security Commander API uses API connections to communicate with Security Commander. There must be one connection record for each type of connection, such as depicted in Figure 1 on page 2.

#### To create an API Connection in Security Commander:

1. If not previously created, use Security Commander's Administration > API Connections form to define each connection record.

Description	Application login
HR Employee Interface	HR data

2. Type a description to identify the connection. The description is used to identify Security Commander API alarms associated with the connection (e.g. HR Employee Data Interface).
3. Type a login name for the connection. You will use this name when configuring Security Commander API for this particular API connection (e.g. HR data).
4. Type a password of at least six characters for the connection (the field displays \* characters). Record the password in a secure location. You will use this password when configuring Security Commander API for this particular API connection.
5. Confirm the password.
6. Click the PC Name arrow and select the Security Commander client or server computer that will be used to host Security Commander API.

7. Click the Operator arrow and select the Security Commander operator authorized to use Security Commander API.
8. Save the API connection record.

### Configuring SPDirWatcher

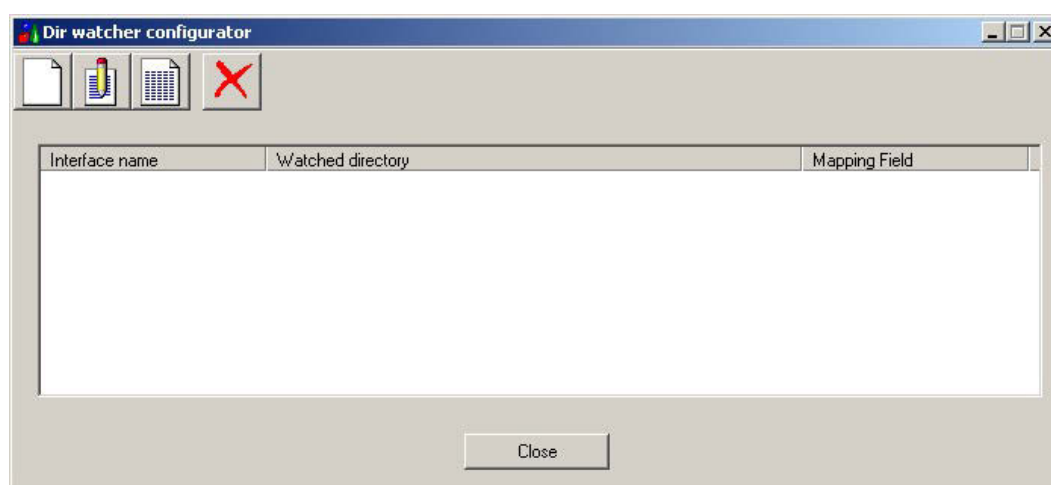
A configuration file SPDirWatcherConfig.xml controls the operation of Security Commander API. This file should be configured using SpDirWatcherConfigurator.exe — an application that defines the:

- Locations of files and folders used by Security Commander API.
- Mapping field for each API connection
- Login name and password for each API connection

#### To configure Security Commander API via the SPDirWatcherConfigurator:

1. In Windows Explorer navigate to the Security Commander folder and run SpDirWatcherConfigurator.exe.

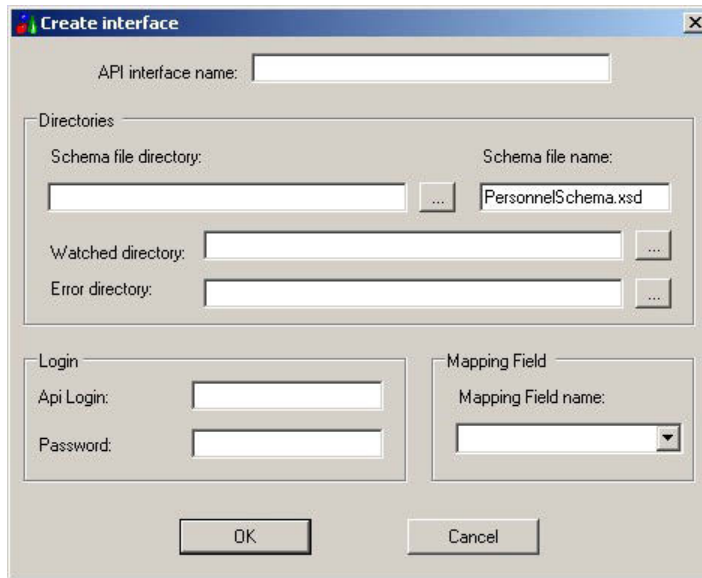
SpDirWatcherConfigurator main window looks as below.



Buttons on the toolbar have following functionality:

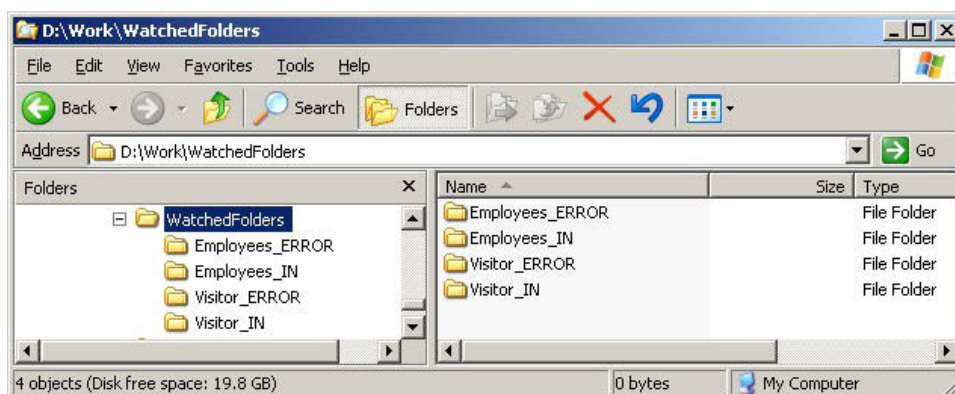
- New: adds new API Connection.
- Edit: edits selected API Connection.
- View: shows details of selected API Connection.
- Delete: removes selected API Connection.

2. Click New button on the toolbar. Following window will appear.



3. In field API Interface Name type the description previously defined in the Security Commander API Connection record, e.g. “HR Employee Data Interface”.
4. In field Schema File Directory type path to directory in which schema file (PersonalSchema.xsd) is located, e.g. C:\Program Files (x86)\UTC Fire & Security\Security Commander\. Use “...” to browse Windows directory tree.
5. In the Watched Directory field type path to the watched folder for the particular API connection, e.g. C:\WatchedFolders\Employees\_IN (every API connection must have a unique watched folder). Use “...” to browse Windows directory tree.
6. In the Error Directory field type path to the error folder for the particular API connection, e.g. C:\WatchedFolders\Employees\_ERRORS (every API connection must have a unique error folder). Use “...” to browse Windows directory tree.
7. In the API Login field type the login name for the connection previously defined in the Security Commander API Connection record, e.g. “HR data”.
8. In the Password field type the password for the connection previously defined in the Security Commander API Connection record.
9. In Mapping Field Name select the Security Commander database field name that will be used to assign each record a unique value. The EmployeeNumber field is typically used as the unique identifier for person records; however, any of the 90 user fields on the Person form can also be used as the unique identifier. Only one API connection may use the EmployeeNumber field as its mapping field — each additional API connection must use its own mapping field selected from the 90 user defined fields.
10. Click OK button to save entered data.
11. If a second API connection is to be used, repeat steps 2 through step 10.

**Note:** It is advised to add all watched and error folders into one common folder (here named “WatchedFolders”).



You must create permissions for each watched folder such that only authorized staff may add, edit, or delete files (refer to Windows documentation for details).

To change parameters of created API connection select it on the list in the main window and click Edit button.

To remove API connection, select it on the list in the main window and click Delete button.

**Note:** SpDirWatcherConfigurator must be used on the same computer as Security Commander API.

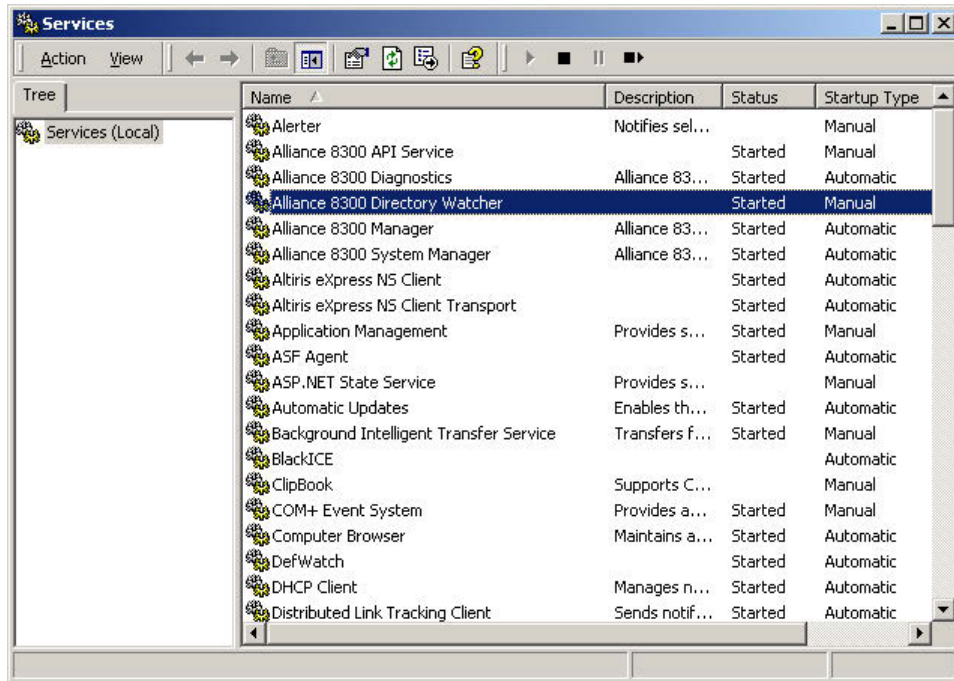
### Starting Alliance 8300 Directory Watcher service manually

In the section “Creating an API Connection in Security Commander” on page 5, you installed the Alliance 8300 Directory Watcher service, which has a default startup type of ‘manual’. When the service is running, Security Commander API will attempt to process any files with a .XML extension that are placed in a watched folder. The service may be left running for as long as you want the processing to occur.

#### To start Alliance 8300 Directory Watcher service manually:

1. Click Start > Settings > Control Panel. Steps for Windows XP vary slightly.
2. Double-click Administrative Tools, and then Double-click Services.

3. Right click the Alliance 8300 Directory Watcher service, and select Start from the popup menu.



The Alliance 8300 API Service starts automatically when the Alliance 8300 Directory Watcher service starts.

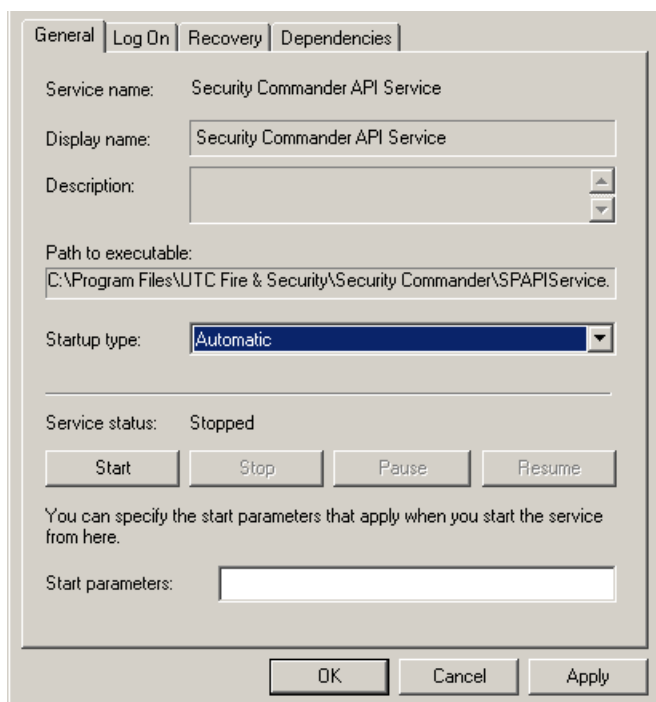
### Starting services automatically

You may wish to change the startup method of the Alliance 8300 Directory Watcher service to automatic. This means every time the computer starts, the Alliance 8300 Directory Watcher service starts.

#### To make services start automatically:

1. Click Start > Settings > Control Panel. Steps for Windows XP vary slightly.
2. Double-click Administrative Tools, and then Double-click Services.

3. Right click the Alliance 8300 Directory Watcher service, and select Properties from the popup menu.



4. Click the Startup type arrow and select Automatic from the list.
5. Click Apply, then click OK to close the window.
6. Close the Services window.
7. Restart the computer to automatically start the Alliance 8300 Directory Watcher service (and the Alliance 8300 API Service).

## Stopping services

When the Alliance 8300 Directory Watcher service is running, Security Commander API will attempt to process any files with a .XML extension that are placed in a watched folder. You may need to prevent this from occurring by stopping the service.

To stop the Alliance 8300 Directory Watcher service:

1. Click Start > Settings > Control Panel. Steps for Windows XP vary slightly.
2. Double-click Administrative Tools, and then Double-click Services.
3. Right click the Alliance 8300 Directory Watcher service, and select Stop from the popup menu.



# Operation

## Overview

In normal operation, Security Commander API can be set to run in the background without Security Commander operator intervention:

- External operators manage person and badge data, and create XML files to update the Security Commander databases with the new data.
- External operators (or applications) copy the XML files to the appropriate watched folder(s) on the Security Commander computer that runs Security Commander API.
- If the Alliance 8300 Directory Watcher service is running, Security Commander API processes XML files as they are received, and changes the Security Commander databases accordingly.
- Changes to Security Commander data (such as a change to a person's badge status) result in downloads to affected Challenger panels in the same manner as if a Security Commander operator made the change.

The Security Commander operator is notified if a fault occurs in this process.

## Person records

Security Commander API may be used to create, update, and delete person and badge records in Security Commander using suitably configured XML files. The files used for this purpose may contain any or all of the three operations.

### Person data

The types of person data that can be managed using Security Commander API include:

- First name
- Last name
- Personnel Type (the personnel type must already exist in Security Commander, refer to *Security Commander Online Help* for details.)
- Employee Number (required if employee number is the designated mapping field)
- Person Number (optional: Security Commander assigns the next higher number if not used)
- Profile Name (the profile must already exist in Security Commander, refer to *Security Commander Online Help* for details.)
- Up to 90 user defined fields (required if a user defined field is the designated mapping field)

## Badge data

The types of badge data that can be managed using Security Commander API include:

- Badge Group (must already exist in Security Commander)
- Number

**Note:** The field contains a standard badge number. PIN or raw card data cannot be used as a badge number.

- Site Code
- Status (Active, Void, Lost, Expired)
- Badge activate date and time (optional)
- Badge deactivate date and time (optional)

## Audit facilities

Record keeping for successful Security Commander API transactions are maintained in Security Commander and may be used to create audit reports.

**To generate a report of changes made to Security Commander database fields by Security Commander API:**

1. Use the Reports > Operator History command to open the Operator History Report form.
2. On the Filters tab, click the Login Name arrow and select the login name for the API connection for which you need a report.
3. Click the Form Name arrow and select <ALL>.
4. On the Date Range tab, select the dates to be reported on.
5. On the Database tab, select either the Security Commander History database or the Security Commander Archive database to cover the selected date range (the archive database settings on the Parameter form determine how long data is kept in the Security Commander History database. The default setting is Daily, which results in data prior to the current date being stored in the Security Commander Archive database).
6. Make other selections for the report, as needed. Refer to *Security Commander Online Help* for additional details.
7. Click the Print Preview button to verify the results of the report before printing.

## Security Commander alarms

Security Commander may detect API-related problems and report these as alarms. These alarms can be used by the operator to investigate faults.

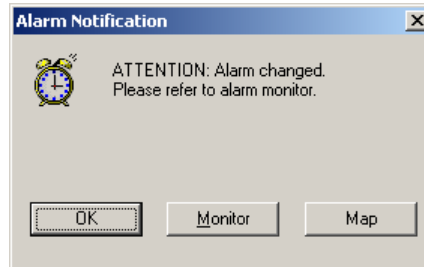
These alarms belong to the owner type API, and include:

- Invalid import file format

- API command failed

When either of these software alarms occurs, the Security Commander operator is notified and details of the alarm are displayed in the Alarm Monitor Form (assuming Monitoring has not been disabled for the alarm type. See *Security Commander Online Help* for details.

**Figure 2: Alarm notification**



The Alarm Monitor Form displays details of the API alarm including the:

- Alarm name.
- Description assigned to the affected API connection (e.g. HR Employee Data Interface).
- Reference details to assist in correcting the fault (e.g. file name of the XML file and reference to the affected record within the file).

The operator may also obtain extended error information from the Security Commander log file.

## Troubleshooting

### Errors folder

Files that cannot be successfully processed are moved to the designated errors folder for the API connection.

This gives the Security Commander operator the ability to edit the file to correct an obvious fault (e.g. if the XML file contains "Standard Profile" when Security Commander contains "StandardProfile" (no space).

After editing, the operator can move the failed XML file back into the appropriate watched folder for the API connection.

### Security Commander API log file

Security Commander API has a log file (SPDirWatcher.log) in the DirWatcher folder, that may be used to investigate problems encountered when processing XML files.

Use a text editor such as Notepad to open and view the SPDirWatcher.log file.

## Security Commander Diagnostic Viewer

If a fault persists with Security Commander API, you may choose to use the Administration > Diagnostic Setting command to enable diagnostic messages for API events.

When set, real-time changes to the appropriate log file can be read in the DiagView window.

**Note:** Continued use of diagnostic messages will degrade the performance of the Security Commander system and consume hard disk space. Refer to *Security Commander Online Help* for details.

# Using XML Files

## Overview

This section describes the structure of XML files for use by Security Commander API. It does not describe how to use external applications to automatically create XML files for use by Security Commander API.

XML files are text files that contain specific elements defined by an XSD file. In the case of Security Commander API, the file is called PersonnelSchema.xsd.

When an XML file is processed by Security Commander API, the defined elements are written to the appropriate records in the Security Commander database. For example, see the following examples of XML files used to create a new Person record:

- The example in Figure 3 below automatically assigns a new, unique person number when creating the record. The new number will be the highest number saved +1.
- The example in Figure 4 on page 16 uses a person number in the "UserNumber" field when creating the record. The person number must not already exist in Security Commander.

**Figure 3: Sample XML file for adding a Person record (without Person Number)**

---

```
<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="PersonnelSchema.xsd">
  <CREATE>
    <NewPerson>
      <FirstName>Gail</FirstName>
      <LastName>Hodgson</LastName>
      <PersonelType>Permanent</PersonelType>
      <EmployeeNumber>1111</EmployeeNumber>
      <ProfileName>standardprofile</ProfileName>
    </NewPerson>
  </CREATE>
</Personnel>
```

**Figure 4: Sample XML file for adding a Person record (with Person Number)**

---

```
<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="PersonnelSchema.xsd">
  <CREATE>
    <NewPerson>
      <FirstName>Gail</FirstName>
      <LastName>Hodgson</LastName>
      <PersonelType>Permanent</PersonelType>
      <EmployeeNumber>1111</EmployeeNumber>
      <ProfileName>standardprofile</ProfileName>
      <UserNumber>44</UserNumber>
    </NewPerson>
  </CREATE>
</Personnel>
```

By virtue of the PersonnelSchema.xsd file (which must be referenced in the XML file), Security Commander API performs the following operations:

- One new person record is created.
- The first name is "Gail".
- The last name is "Hodgson".
- The personnel type is "Permanent" (this value must already exist in Security Commander).
- The employee number is "1111" (when creating records this value must not already exist in Security Commander)
- The Profile name is "StandardProfile" (this value must already exist in Security Commander).
- The optional person number is "44" (when creating records this value must not already exist in Security Commander).

The Security Commander operations that may be performed using XML files are:

- Create: Add a new person record and badge record(s).
- Update: Change details of an existing person record, including badge details.
- Delete: Remove a person record and/or badge record(s).

It makes no difference whether these operations are placed in separate XML files or combined into a single file.

## Creating new person records

### Example 1 — New person

Refer to the Figure 1 on page 2 for adding a new person record. The following rules apply.

**Table 3: Rules for new person records**

Element	Required	Comments
FirstName	Yes	Up to 32 characters of text.
LastName	Yes	Up to 32 characters of text.
PersonelType	Yes	Must already exist in Security Commander. Up to 32 characters of text.
EmployeeNumber	Yes	Up to 12 characters of text.
UserNumber	No	Up to 6 digits.
ProfileName	No	Up to 50 characters of text.
Badges	No	If used, must contain further elements, indicated by indented text. Refer to Table 4 on page 18 for details of these further elements.
UserDefinedFields	No	Must be used where specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection. If used, must contain further elements, indicated by indented text. Refer to Table 5 on page 19 for details of these further elements.

### Example 2 — New person with badge

Figure 5 below depicts an XML file similar to Figure 3 on page 15, but containing details of a badge (a person record may contain more than one badge).

The details of the badge are indicated in italic (not used in XML files).

**Figure 5: Sample XML file for adding a person record containing a single badge**

```
<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">
  <CREATE>
    <NewPerson>
      <FirstName>Gail</FirstName>
      <LastName>Hodgson</LastName>
      <PersonelType>Permanent</PersonelType>
      <EmployeeNumber>1111</EmployeeNumber>
```

```

    <ProfileName>standardprofile</ProfileName>
    <Badges>
      <Badge>
        <GroupName>TestGroup</GroupName>
        <Number>171</Number>
        <SiteCode>123</SiteCode>
        <Status>Active</Status>
      </Badge>
    </Badges>
  </NewPerson>
</CREATE>
</Personnel>

```

The following rules apply to badge elements.

**Table 4: Rules for badge elements**

Element	Required	Comments
GroupName	Yes	Must already exist in Security Commander. Up to 50 characters of text.
Number	Yes	Must be an integer.
SiteCode	Yes	Must be an integer.
Status	Yes	Must be one of Active, Expired, Lost, or Void.
Activate	No	If used, must be in date time format YYYY-MM-DDTHH:MM:SS
Deactivate	No	If used, must be in date time format YYYY-MM-DDTHH:MM:SS

### Example 3 — New person with user defined field mapping

Figure 6 below depicts an XML file similar to Figure 3 on page 15, but where a user defined field (e.g. UserDefinedField1) is specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection instead of using the EmployeeNumber field.

In the following figure, the details of the user-defined field are indicated in *italic* (not used in XML files).

**Figure 6: Sample XML file for adding a person record where UserDefinedField1 is the unique mapping field**

```

<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">
  <CREATE>

```



```

    <NewPerson>
      <FirstName>Gail</FirstName>
      <LastName>Hodgson</LastName>
      <PersonelType>Permanent</PersonelType>
      <EmployeeNumber>1111</EmployeeNumber>
      <ProfileName>standardprofile</ProfileName>
      <UserDefinedFields>
        <UserDefinedField1>OIV871</UserDefinedField1>
      </UserDefinedFields>
    </NewPerson>
  </CREATE>
</Personnel>

```

The XML file can contain up to 90 user defined fields (on separate lines) between the <UserDefinedFields> line and the </UserDefinedFields> line, any one of which may be defined as the unique mapping field.

Any one of the 90 user defined fields in Security Commander may be used as the MappingFieldName. The following rules apply.

**Table 5: Rules for User Defined Field elements**

Element	Required	Comments
UserDefinedField1 through UserDefinedField90	No*	Up to 32 characters of text

\* If the SPDirWatcherConfig.xml file for the API connection requires a particular user defined field to be the mapping field, then the field is required to be used.

## Updating Person records

The rules for updating person records are simpler than the rules for creating person records because less information is required (there are fewer elements marked 'Yes').

### Example 1 — Updated person record

The following example shows an XML file used to change the Personnel Type of a Person record to "Temporary".

**Figure 7: Sample XML file for updating a Person record**

```

<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">
  <UPDATE>

```

```

    <Person>
      <PersonelType>Temporary</PersonelType>
      <EmployeeNumber>JKFF-85981</EmployeeNumber>
    </Person>
  </UPDATE>
</Personnel>

```

Other than the mapping field, only the changed data must be included.

Refer to the Figure 7 on page 19 example of an XML file for updating a Person record. The following rules apply.

**Table 6: Rules for updating person records**

Element	Required*	Comments
FirstName	No	Up to 32 characters of text.
LastName	No	Up to 32 characters of text.
PersonelType	No	Must already exist in Security Commander. Up to 32 characters of text.
EmployeeNumber	Yes	Up to 12 characters of text. Not required if a user defined field is specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection.
UserNumber	No	Up to 6 digits.
ProfileName	No	Up to 50 characters of text.
Badges	No	If used, must contain further elements, indicated by indented text. Refer to Table 4 on page 18 for details of these further elements.
UserDefinedFields	No	Must be used where specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection. If used, must contain further elements, indicated by indented text. Refer to Table 5 on page 19 for details of these further elements.

\* Any relevant data that has been changed is required.

## Example 2 — Updated person and badge records

The following example shows an XML file used to change a Badge from “active” to “void”. In the following figure, the details of the badge are indicated in *italic* (not used in XML files).

**Figure 8: Sample XML file for updating a person’s badge record**

```

<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">

```

```

<UPDATE>
  <Person>
    <EmployeeNumber>JKFF-85981</EmployeeNumber>
    <Badges>
      <Badge>
        <GroupName>TestGroup</GroupName>
        <Number>171</Number>
        <SiteCode>123</SiteCode>
        <Status>Void</Status>
      </Badge>
    </Badges>
  </Person>
</UPDATE>
</Personnel>

```

The update function may be used to add badges to a person record. For example, if someone needs a second badge, you would use an XML file containing an UPDATE section to add the second badge to the person record (identified by the EmployeeNumber or other matching field, as applicable).

Security Commander API automatically creates the new badge in Security Commander.

## Deleting persons or badges

Security Commander API allows deletion of persons and/or badges.

Any badges that are assigned to the person when the person record is deleted, Security Commander API automatically sets the badges to “unassigned” and “void”, and the badges are removed from the relevant Challenger hardware.

### Example 1 — Deleting a person record

For example, see the following example of an XML file used to delete a person record.

**Figure 9: Sample XML file for deleting a person record. Only the mapping field must be included**

---

```

<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">
  <DELETE>
    <Person>

```

```

        <EmployeeNumber>JKFF-85981</EmployeeNumber>
    </Person>
</DELETE>
</Personnel>

```

Refer to the Figure 9 on page 21 for an example of an XML file for deleting a person record. The following rules apply.

**Table 7: Rules for deleting person records**

Element	Required	Comments
FirstName	No	Up to 32 characters of text.
LastName	No	Up to 32 characters of text.
PersonelType	No	Must already exist in Security Commander. Up to 32 characters of text.
EmployeeNumber	Yes	Up to 12 characters of text. Not required if a user defined field is specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection.
UserNumber	No	Up to 6 digits.
ProfileName	No	Up to 50 characters of text.
Badges	No	If used, must contain further elements, indicated by indented text. Refer to Table 4 on page 18 for details of these further elements.
UserDefinedFields	No	Must be used where specified as the MappingFieldName by the SPDirWatcherConfig.xml file for the API connection. If used, must contain further elements, indicated by indented text. Refer to Table 5 on page 19 for details of these further elements.

## Example 2 — Deleted a badge record

For example, see the following example of an XML file used to delete a badge record.

**Figure 10: Sample XML file for deleting a badge record**

```

<?xml version="1.0" encoding="utf-8"?>
<Personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noN-
amespaceSchemaLocation="PersonnelSchema.xsd">
    <DELETE>
        <Badge>
            <GroupName>TestGroup</GroupName>

```

```

        <Number>171</Number>
        <SiteCode>123</SiteCode>
    </Badge>
</DELETE>
</Personnel>

```

Refer to the Figure 10 on page 22 example of an XML file for deleting a badge record. The following rules apply.

**Table 8: Rules for deleting badge records**

Element	Required	Comments
GroupName	Yes	Must already exist in Security Commander. Up to 50 characters of text.
Number	Yes	Must be an integer.
SiteCode	Yes	Must be an integer.

**Note:** No other fields may be used when deleting badges.

## File naming and handling

The requirements for Security Commander API to process XML files are:

- The files are correctly constructed (according to PersonnelSchema.xsd).
- The file extension is .xml

All successfully-processed files are deleted by Security Commander API after processing. As a result, it is the customer's responsibility to keep a copy of XML files if such record-keeping is required.

It is advisable to adopt a file-naming convention such that file names are unique and the file name relates to the particular API connection (if applicable). This could help when troubleshooting faults.

