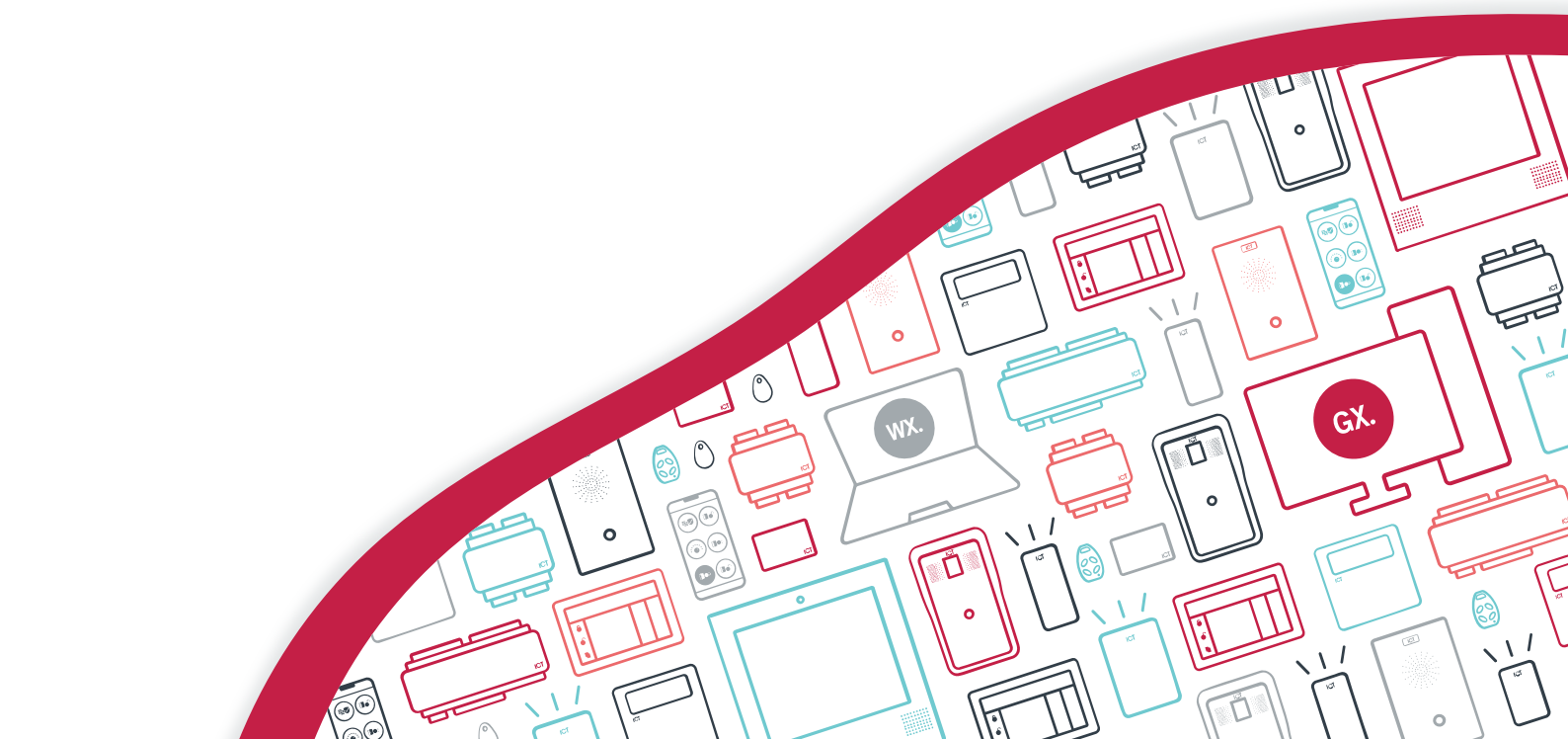
AN-218

Configuring Credential Types in Protege WX

Application Note



Last Published: 18-Oct-23 2:12 PM

Contents

Introduction

Credential types are used by the Protege WX system to recognize alternative forms of user identification for the purpose of access control. They enable a controller or reader expander to interpret custom Wiegand card formats from card readers connected to the reader ports. The controller can also recognize credential data in a wide variety of other formats, received through either the ethernet port or RS-485 reader port (often via an RS-232 to RS-485 converter). This provides a flexible method of integrating with third-party systems and card readers.

This feature has a broad range of applications, including:

- The use of uncommon or legacy card formats in Protege WX.
- License plate recognition via video systems connected over IP.
- Integration with third-party modules which send data such as PINs or passwords over serial or ethernet interfaces.

This application note provides instructions on programming Protege WX for operation with custom credential types.

This document only covers the programming that is relevant to Protege WX. Third-party applications and devices may have their own programming requirements.

Prerequisites

Programming credential types in Protege WX requires access to Advanced Mode.

Component	Version
Protege WX Controller	4.00.649 or higher.

Licensing Requirements

License	Ordering Code	Notes
Protege WX Third-Party Reader Interface License	PRT-WX-TPR-IF	1 license required for each smart reader record configured to utilize the credential type for door entry or exit.

Programming

Creating a Credential Type

The credential type programming defines the format of the data that will be received from the third-party devices or application.

1. Navigate to **Users | Credential Types** and click **Add** to create a new credential type.
2. Assign an appropriate **Name**.
3. In the **Configuration** section select the **Format** of the data to be sent to the Protege WX controller from the third-party device or application. Supported formats include:
   - **Unicode**: The credential data sent to the controller uses two bytes to represent each character as per the Unicode standard.
   - **UTF8**: The credential data sent to the controller uses a variable number of bytes to represent each character as per the UTF-8 standard.
   - **ASCII**: The credential data sent to the controller uses a single byte to represent each character as per the ASCII standard.
   - **Numeric**: The credential data sent to the controller is a binary number composed of up to 8 bytes. The bytes are ordered using little endian.
   - **Hexadecimal**: The credential data is sent to the controller as an array of binary numbers. When the specific credential is entered into the user programming for each user, the format used is hexadecimal with the numbers 0-9 and letters A-F representing each nibble of the credential.
   - **Wiegand**: The credential data sent to the controller is composed of a Wiegand bit stream.

     This bit stream can be encoded in numerous different ways and a format descriptor must be included in the **Wiegand or TLV Format** field. For the Wiegand format the preceding, trailing and prefix character settings and case sensitive setting are ignored.

   Controllers support all credential type formats via either RS-485 or ethernet. Reader expanders only support Wiegand credential types.
4. If you have selected **Wiegand**, enter the **Wiegand or TLV Format**. This is a string that describes how each bit of the Wiegand card data should be read. For more information, see Appendix: Configuring Custom Wiegand Credential Types (page 9).
5. Enter the number of **Preceding Characters** included at the start of the credential data sent to the controller. This number represents the maximum number of characters that may be present in the received data before the actual credential is found. If the credential is found before this number of characters are counted then the credential will still be accepted.

   This setting is determined by the third-party device/application.
6. Enter the number of **Trailing Characters** included at the end of the credential data sent to the controller. This number represents the maximum number of characters that may be present in the received data after the actual credential is found. If the credential is found and there are fewer than this number of trailing characters then the credential will still be accepted. For example, this can be used to accept a credential whether or not it is followed by a carriage return character.

   This setting is determined by the third-party device/application.
7. Enter the required **Prefix** characters. These are the characters that are required in the data string immediately before the user specific credential data.

   This setting is determined by the third-party device/application.
8. Select the **Case Sensitive** option if required for this data configuration.
9. Click **Save**.

Assigning the Credential to Users

Once a credential type is added it becomes available to assign to users. In order to utilize the credential you need to enter each user's unique credential details for the credential type, so that the user can be identified.

1. Navigate to **Users | Users**.
2. In the **Credentials** tab, click **Add** and select the required **Credential Type** from the dropdown.
3. Enter the user's unique identification details in the corresponding **Credential** field.

   The user's identification details will often be generated by the third-party device or application.

4. Click **Save**.

Onboard Reader Expander Configuration

If data is sent from the third-party source over ethernet, this can only be received via the controller's onboard reader expander which will need to be configured to receive the data.

The controller can also be configured to receive data via connection to its onboard RS-485 reader ports.

1. Navigate to **Expanders | Reader Expanders** and select the controller's onboard reader expander.
2. Set the following **Configuration** network options:
   a. If connection to the controller is via onboard RS-485 reader port:
      - **Port Network Type**: for the port the device is physically connected to, set the **Port Network Type** to Third Party Generic.
      - **Ethernet Network Type**: Disabled
      - Select the appropriate Reader tab for the port in use (**Reader 1 or Reader 2**), scroll down to the **Third Party Generic** section and set the following options:
        - **Baud Rate**: The rate at which information is transferred between the third-party device and the Protege controller.
        - **Parity**: Defines the method of calculating the parity for the block
        - **Stop Bits**: Defines the number of stop bits used.
        - **Inter-Byte Time Out**: Defines the maximum time waited, in milliseconds, between receiving each byte. When this time is exceeded the captured bytes are processed as a packet.
        - **Log Invalid Data Received**: When enabled, instances where invalid data is received are logged.

      If the third-party device uses an RS-232 connection, you must use an RS-232 to RS-485 converter to connect it to the Protege reader expander or controller.

   b. If connection to the controller is over ethernet:
      - **Ethernet Network Type**: Third Party Generic
      - **Ethernet Port**: Defines the first TCP/IP port the controller listens on.

        This port must be open in order to send data to the controller.

3. If the controller needs to listen for multiple ethernet connections, Smart Reader Port Offset must also be enabled. In the **Commands** field on the **General** tab add the command:

   ```
   SmartReaderPortOffset = true
   ```

   This command allows the controller to receive ethernet data over a range of ports above the one set above. The controller will listen on every port equal to the **Ethernet Port** + the **Configured Address** of each smart reader associated with the ethernet port. This allows the data from each device to be sent to the controller over a unique TCP/IP port so that the controller can distinguish between them.

4. Click **Save**.

Configuring Reader Expanders

Reader expanders can recognize alternative forms of user identification from connected third-party devices. To enable this the reader expander will need to be configured to receive credential data from the connected device as a custom credential.

This configuration only applies when the devices reading the custom credentials are physically attached to reader expanders (e.g. third-party proximity readers). This does not include devices which send credential data over ethernet, such as cameras linked to a DVR/NVR.

1. Navigate to **Expanders | Reader Expanders** and select the reader expander the device is connected to.
2. For the port the device is physically connected to, set the **Port Network Type** to Third Party Generic.

   This sets the reader format to custom credential. Other format settings are ignored.
3. Click **Save**.
4. Select the appropriate **Reader** tab for the port in use and set the **Reader Mode** to Access so the reader can be used for reading the third-party device.
5. Click **Save**.
6. Repeat the above steps for every reader expander that has a connected device using this credential type.

With the Port Network Type set to Third Party Generic the Secondary Format is not applicable. To implement additional formats use Fallback Door Types as described in Creating a Custom Door Type (see next page).

Creating Smart Readers

A smart reader represents the third-party reading device or software that you are connecting to, and provides the programming to link the credential type functionality to door operation. For each door entry and exit that will utilize the credential type you will need one smart reader record configured as a third-party reader.

One Third Party Reader license (ordering code: PRT-WX-TPR-IF) is required for each smart reader record that is configured to utilize the credential type for door entry or exit.

1. Navigate to **Expanders | Smart Readers** and click **Add**.
2. Assign a **Name** that identifies the connected device and the door it controls.
3. Set the **Expander Address** to the Physical Address of the reader expander the device is connected to.
4. Set the **Expander Port** to the port the device is connected to.
5. Set the **Configured Address** to any available unique address. This value must not be left as <not set>.

   Note: If the `SmartReaderPortOffset = true` command has been entered in the onboard reader expander programming:
   - The **Configured Address** of each smart reader is added to the **Ethernet Port** configured on the reader expander. The controller will open these ports to listen for incoming credentials.

     For example, if the **Ethernet Port** on the reader expander is set to 9000 and two smart readers are programmed at **Configured Address** 9 and 10, the controller will be listening on ports 9009 and 9010.

     The controller will listen at both ports 9000 and 9001 for the smart reader programmed at **Configured Address** 1.
6. Select the **Reader** tab and assign the **Reader 1 Door** to specify which door the smart reader will unlock.
7. If the third-party device is physically connected to a reader expander RS-485 port, select the **Reader 1 Format** that the connected device will use to send credential data.

8. If the third-party device will send credential data over ethernet, select the **Reader Credential Types** tab and click **Add**, then select the required credential type and click **OK**.

   The Reader Credential Types tab will only be available when the **Ethernet Network Port** of the assigned reader expander (determined by the **Expander Address** selection) is set to Third Party Generic.

9. Click **Save**.
10. Repeat the above steps for every smart reader required to control doors with this credential type.

Creating a Custom Door Type

A custom door type is required to instruct the door to respond to the defined credential type.

1. Navigate to **Programming | Door Types** and click **Add**.
2. Assign a **Name**.
3. To use the credential type for entry set the **Entry Reading Mode** to Custom.
4. In the **Entry Credential Types** tab, click **Add** to define the valid credential type(s) for this door type.
5. Select the credential types required for entry and click **OK** to add.

   Including more than one entry credential type specifies that all of those credentials are required to gain access to the door. To specify an alternative credential type, assign a **Fallback Door Type**.

6. If users are required to present multiple credential types in a specific order on entry, enable the **Sequence** option and use the arrows to set the sequence order.
7. To use the credential type for exit set the **Exit Reading Mode** to Custom.
8. In the **Exit Credential Types** tab, click **Add** to define the valid credential type(s) for this door type.
9. Select the credential types required for exit and click **OK** to add.
10. If users are required to present multiple credential types in a specific order on exit, enable the **Sequence** option and use the arrows to set the sequence order.
11. A more complex setup may require an alternative set of credentials to be accepted. To accommodate this a **Fallback Door Type** can be assigned in the **General Configuration** section. For example, a door type that requires card and biometric could be configured with a fallback door type that requires card and PIN. In this case presenting either card and biometric or card and PIN credentials will grant access to the door.

    The fallback door type can itself have a fallback door type. This allows for a number of credential types to be configured for any one door.

12. Click **Save**.

Assigning the Door Type to Doors

Every door to be controlled using the defined credential type(s) needs to be assigned the custom door type created above.

1. Navigate to **Programming | Doors** and select a door that will allow access with the new credential type.
2. Set the **Door Type** to the custom door type created above.
3. Click **Save**.

Repeat for every door that will allow access using the new credential.

Appendix: Configuring Custom Wiegand Credential Types

The Wiegand credential type allows you to program custom card formats, enabling the controller or reader expander to recognize card formats which are not included in the standard programming.

The **Custom reader format** tab in the controller programming also allows you to program custom card formats, however this only allows one custom format to be configured per controller. The credential types feature allows you to program multiple custom card formats.

To create this credential type, you must enter a **Wiegand or TLV format** to define how incoming data should be interpreted. The example below shows how to define the standard HID 34 bit format.

```
# 34bit
#
# Variables
A,FACILITY,16,MSB,BIN
B,CARD,16,MSB,BIN
# Format
PAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBP
# Parity
EXXXXXXXXXXXXXXX.................
.................XXXXXXXXXXXXXXXO
```

Notes on the format definition:

- Each line in the format definition should be separated by a single or double underscore. For this example, you would enter:

  ```
  #34bit__A,FACILITY,16,MSB,BIN__B,CARD,16,MSB,BIN__
  PAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBP__EXXXXXXXXXXXXXXX.................._
  _.................XXXXXXXXXXXXXXXO
  ```

- Lines beginning with **#** are for human readability only and are ignored by the controller.
- The **Preceding characters**, **Trailing characters**, **Prefix** and **Case sensitive** fields are not applicable for this format.

Variables

Variables must be defined for both the facility and card codes. In this example:

```
A,FACILITY,16,MSB,BIN
```

- The first value (**A**) is the symbol that is used in the format definition to describe the relevant bit positions. The symbol can be anything other than **#**, **P**, **X**, **.** or **,** but would typically be an early letter of the alphabet.
- The second value (**FACILITY**) defines whether the symbol refers to a facility or card code. Valid values are **FACILITY** or **CARD**.
- The third value (**16**) defines the number of bits that make up the code.
- The fourth value (**MSB**) defines whether the run of bits will be interpreted most significant bit first or least significant bit first. Valid values are **MSB** (most significant bit first) or **LSB** (least significant bit first).
- The fifth value (**BIN**) defines the structure of the bit pattern and whether it should be interpreted as a binary number or as a binary coded decimal number. Valid values are **BIN** (binary number) or **BCD** (binary coded decimal).

Format

The format defines where bits are located in the Wiegand bit stream. In this example:

```
PAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBP
```

- Each letter corresponds to a single bit that is received in the Wiegand bit stream.
- **P** denotes a parity bit.
- **A** or **B** denotes a bit corresponding to one of the previously defined **variables**. The letter used will depend on the symbol defined for each variable.
- In the example shown above, the bit stream is defined as having one parity bit, followed by 16 facility code bits, followed by 16 card code bits, followed by one parity bit.

Parity

Finally, the parity bits are defined. For example:

```
EXXXXXXXXXXXXXXXX.................
```

- Each letter/symbol corresponds to a single bit that is received in the Wiegand bit stream.
- **O** or **E** defines the position of an **odd** or **even** parity bit.
- **X** defines the position of a bit to be included in the parity calculation.
- **.** defines the position of a bit that is not included in the parity calculation.
- Multiple lines may be required to cover multiple parity bits.

Including more than one **O** or **E** in a single line of parity calculation definition will result in incorrect interpretation of the bit stream.

Fixed Bits

- If the card contains fixed (always present) bits then these can be defined in a similar way to the card and facility bits using a third entry:

  ```
  C,FIXED,4,MSB,BIN
  ```

  The number, **4** in this case, is the decimal value of the fixed bit stream.

  The following example defines the **Infinity 37 bit** Wiegand format which contains the fixed bits **010** in the second, third and fourth bit positions. The binary value **010** equals the value **4** in decimal notation.

  ```
  #Infinity37bit__A,FACILITY,12,MSB,BIN__B,CARD,19,MSB,BIN__
  C,FIXED,4,MSB,BIN__PCCCCAAAAAAAAAAAABBBBBBBBBBBBBBBBBP_
  _.X.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.E__
  O.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.X.
  ```

- Alternatively, fixed bits can be represented directly in the format string. An alternate representation of the Infinity 37 bit Wiegand format would be:

  ```
  #Infinity37bit__A,FACILITY,12,MSB,BIN__B,CARD,19,MSB,BIN__
  P0100AAAAAAAAAAAABBBBBBBBBBBBBBBBBP_
  _.X.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.E__
  O.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.XX.X.
  ```

Configuring a Descramble Format

Some proprietary 125kHz credentials send card data in a scrambled order. This is common with PSK (phase-shift keying) card formats and makes it difficult to transition legacy sites which use these credentials to new card readers. However, you can program a custom Wiegand credential type which allows the controller to interpret these credentials even if the card reader cannot.

Contact ICT Technical Support for assistance with this feature.

Designers & manufacturers of integrated electronic access control, security and automation products.
Designed & manufactured by Integrated Control Technology Ltd.