

AxoNet Plug-Ins for InstallAware

Quick reference guide

Edition 2.2

Contents

1. About	2
2. Upgrade Code – Plug-In.....	3
2.1. Usage.....	3
3. Volume Info – Plug-In	4
3.1. Usage.....	4
3.2. Note.....	5
4. Windows Firewall – Plug-In	6
4.1. Usage.....	7
4.1.1. Adding a program	7
4.1.2. Adding a port and protocol	7
4.1.3. Deleting a program	8
4.1.4. Deleting a port	8
4.2. Requirements.....	9
5. Revision history	10

1. About

AxoNet Plug-Ins for InstallAware is a Freeware Add-On package for InstallAware 6.0 – 12.0. This software is provided as is, use on your own risk. Liability for any damage due to faulty software or data is impossible.

The plug-ins have been developed with CodeGear RAD Studio 2007 (Delphi) and tested with InstallAware 6.1 and InstallAware 12.0

Copyright © 2006-2011, AxoNet Software GmbH, Martin Rothschink

2. Upgrade Code – Plug-In

This plug-in detects a product based on the upgrade code. It returns the product code if the upgrade code is found. This is useful if you upgraded from InstallShield Express to InstallAware. InstallShield Express requires a change of the product code for every new version of your product and keeps the upgrade code always the same.

Example

Product 1.0 Product Code A, Upgrade Code U

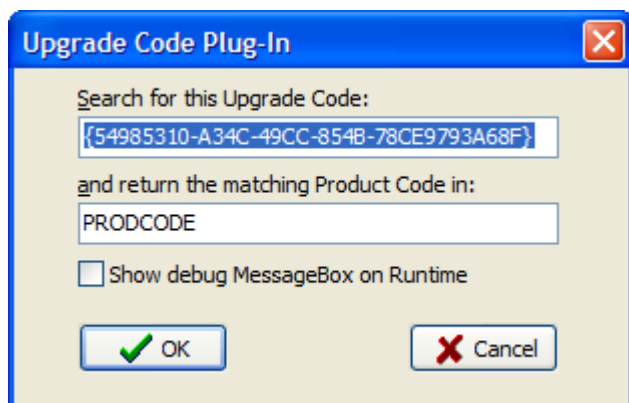
Product 1.1 Product Code B, Upgrade Code U

Product 1.2 Product Code C, Upgrade Code U

InstallAware never changes the product code; instead the revision code is changed. If a new version is installed, a previous version is detected by the same product code and removed. If you have previously used InstallShield Express for your product, you do not know which product code is present on the target machine. Use this plug-in to detect and uninstall a previous version.

2.1. Usage

Go to the “Check Application Pre-Requisites” section. Define a variable which will store the product code. Add this plug-in and specify your InstallShield Express upgrade code (you may also use a variable which holds the GUID):



Check if a product code is returned and uninstall the old product:

```
[DEFINE REGION: Check Application Pre-Requisites]
...
Set Variable PRODCODE to
Get Product Code from Upgrade Code {54985310-A34C-49CC-854B-78CE9793A68F} into PRODCODE
if Variable PRODCODE not Equals
    Install/Remove MSI Package $PRODCODE$[REMOVE=ALL]
end
...
```

3. Volume Info – Plug-In

This plug-in retrieves information about a logical drive. You can specify a single drive letter, a root directory, a full path name or an UNC share name.

Examples

C
C:\
C:\path\file
\\server\share

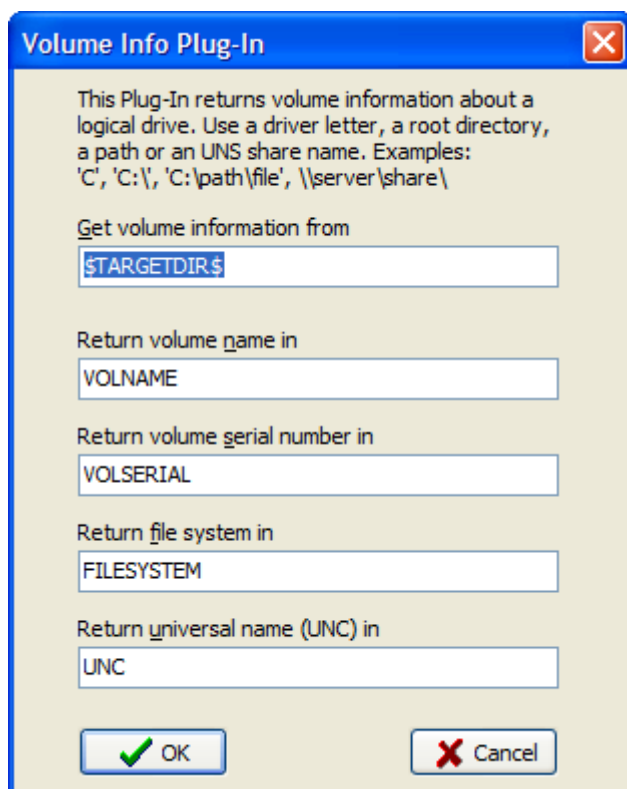
The plug-in returns the volume name, serial number, file system type and the UNC path name.

The file system type is one of

FAT
FAT32
NTFS
CDFS

3.1. Usage

Add the plug-in to your script where you need to retrieve volume information. Define variables to hold the retrieved information:



AxoNet Plug-Ins for InstallAware

Quick reference guide

```
Set Variable VOLNAME to  
Set Variable VOLSERIAL to  
Set Variable FILESYSTEM to  
Set Variable UNC to  
Get Volume Information from '$TARGETDIR$'
```

3.2. Note

If the volume is not a network share, UNC is always empty.

4. Windows Firewall – Plug-In

This plug-in configures the Windows Firewall. You can add or delete a firewall exception based on an executable program or a port number and protocol.

Example MSI code for adding exception rules

To successfully add a program to the firewall list, the program must exist! Therefore you should always place the “Configure Windows Firewall” plug-in after the Apply Install command in your MSI code:

```
if Variable ADVERTISE Equals TRUE
  Apply Advertised (get result into variable SUCCESS)
else
  Apply Install (get result into variable SUCCESS)
end
```

```
Configure Windows Firewall - add allowed program $TARGETDIR$\NOTEPAD.EXE
Configure Windows Firewall - add port opening 1234 Both
```

```
[compiler end]
Set Variable PROGRESS to 100
```

Example MSI code for deleting exception rules

This is typically done on uninstall. Place the “Configure Windows Firewall” plug-in after the TO-DO comment.

```
Comment: Modify Target System
[DEFINE REGION: Perform Uninstallation]
if Variable REMOVE Equals TRUE
  Comment: Uninstall product
  Comment: TO-DO: Insert any additional uninstall commands here
```

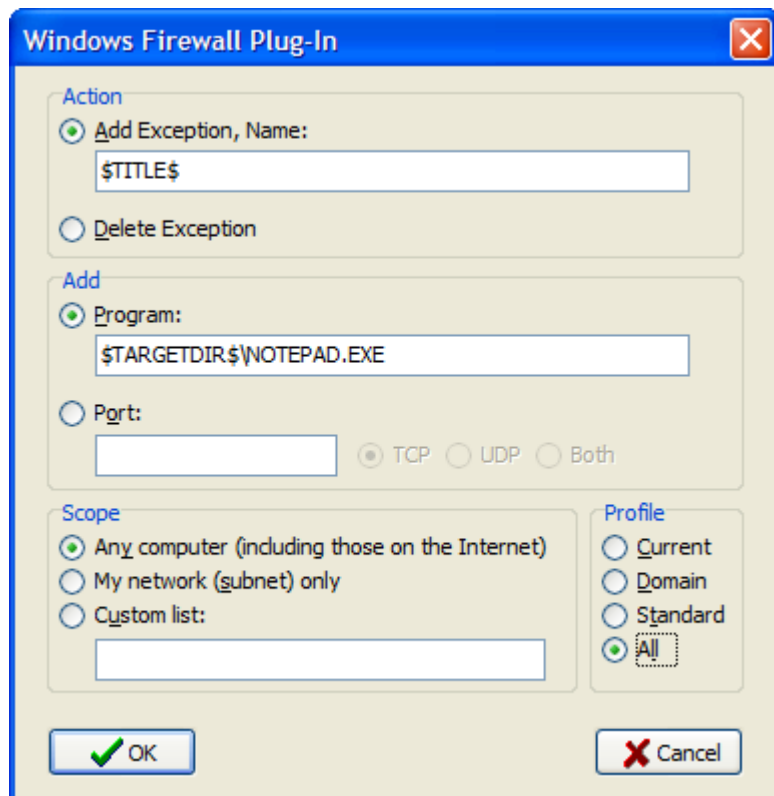
```
Configure Windows Firewall - delete allowed program $TARGETDIR$\NOTEPAD.EXE
Configure Windows Firewall - delete port opening 1234 Both
```

```
Apply Uninstall (get result into variable SUCCESS)
Set Variable PROGRESS to 100
```

4.1. Usage

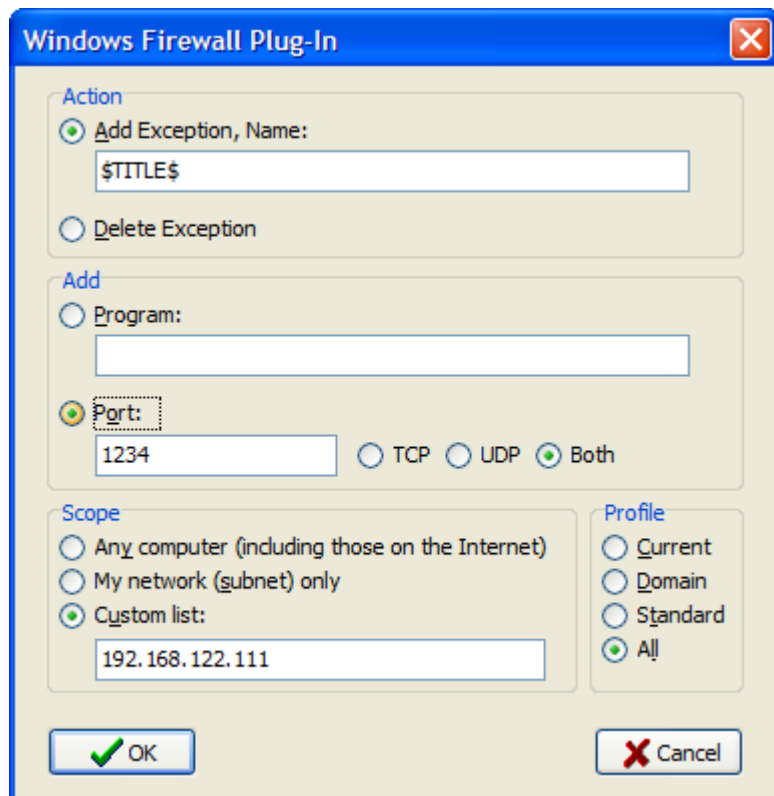
4.1.1. Adding a program

To add a program, enter an exception name and the full program path. You can use variables for both fields. The exception name is displayed in the Windows Firewall applet after installation. Optionally you may modify the scope and the profile:



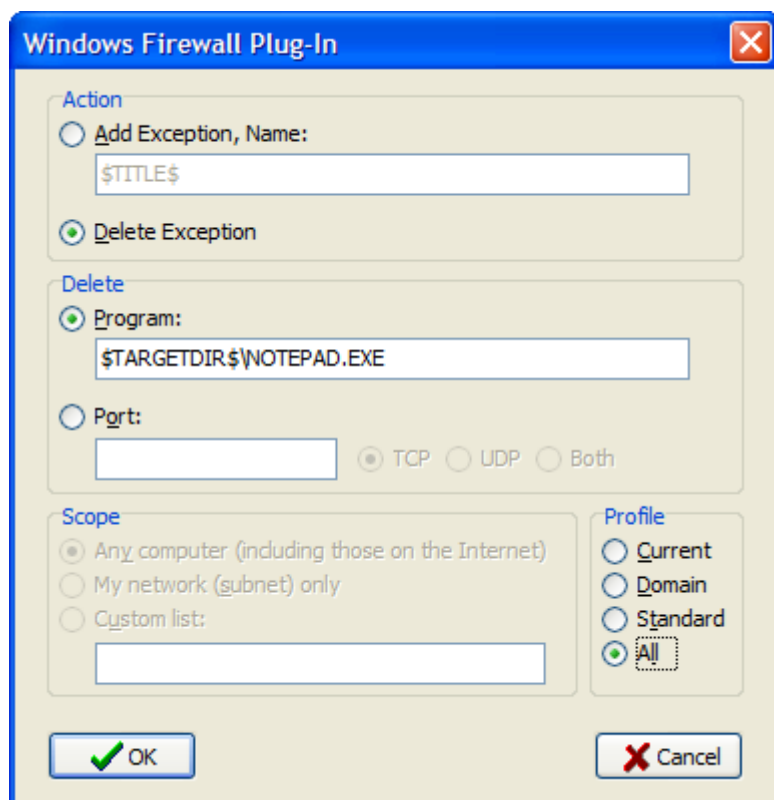
4.1.2. Adding a port and protocol

To add a port number and protocol you have to enter an exception name, the port number and the protocol. If you select "Both", two entries are created, one for UDP and one for TCP:



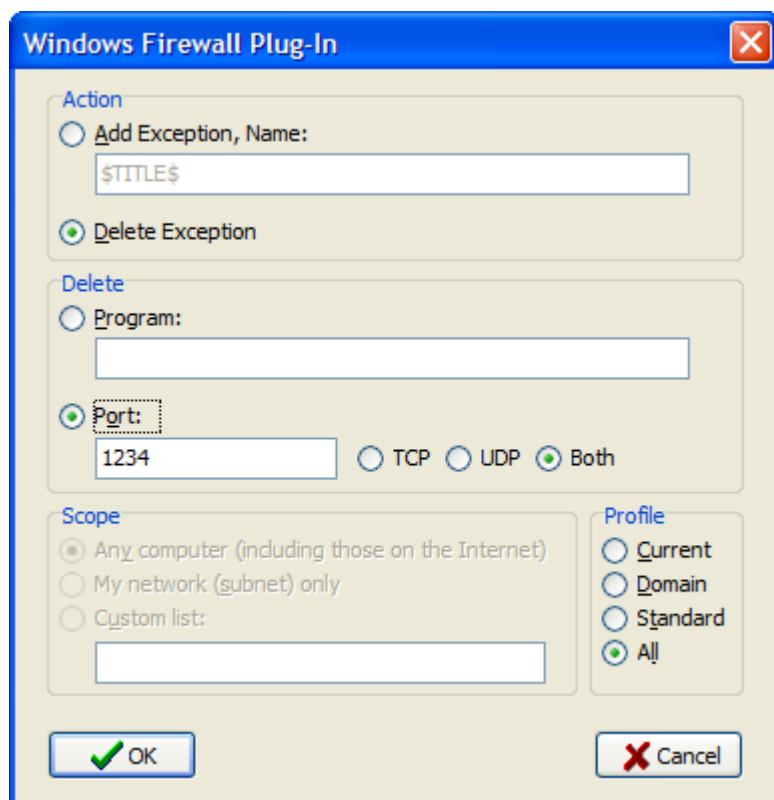
4.1.3. Deleting a program

To delete a program from the exception list you only need to specify the full program path:



4.1.4. Deleting a port

To delete a port from the exception list, specify the port number and protocol:



4.2. Requirements

This plug-in requires Windows XP SP2 or greater. You should wrap the “Configure Windows Firewall” plug-in command in an “if ... end” section if you create setups for other operating systems.

5. Revision history

Version 2.2

- Fixed UNC path problem in Firewall – Plug-In

Version 2.1

- Added UNC path to Volume Info - Plug-In
- Fixed x64 problem in Upgrade Code - Plug-In

Version 2.0

- Added Windows Firewall - Plug-In

Version 1.0

- First public release