



DoD Public Key Enablement (PKE) Reference Guide

TACT v1.2.6 User Guide

Contact: [dodpke@mail.mil](mailto:dodpke@mail.mil)

URL: <http://iase.disa.mil/pki-pke>

Enabling PKI Technology  
for DoD users

# Trust Anchor Constraint Tool (TACT) v1.2.6 User Guide

2 September 2015

Version 1.2.6

DoD PKE Team

## Revision History

Issue Date	Revision	Change Description
01/27/2012	1.0	Initial release
4/4/2012	1.0.2	Add back-out plan appendix and reference to TACT Operator utility
10/24/2012	1.1.0	Added content for new country code filtering support, batched TAMP message processing support and resource-focused configuration testing.
1/27/2013	1.1.1	Addressed internal review comments
3/20/2013	1.1.2	Added Win 8 and Server 2012 information, bumped TACT versions
6/16/2014	1.2.0	Documented new features for mod_nss TACT variants
6/26/2014	1.2.0b	Fixed broken ISO link
8/15/2014	1.2.2	Adding missing screen shot and text for a panel in security environment
1/15/2015	1.2.5	Removed reference to Windows 2003 and Apache 2.0 as these are no longer supported
09/02/2015	1.2.6	Updated version information.

# Contents

<b>OVERVIEW .....</b>	<b>1</b>
BACKGROUND .....	1
INTENDED AUDIENCE .....	2
SUPPLEMENTAL INFORMATION .....	2
<b>TACT OVERVIEW .....</b>	<b>3</b>
TACT COMPONENTS .....	3
TACT DATA .....	4
<i>Server configuration</i> .....	5
<i>Logging settings</i> .....	6
<i>PKI Log</i> .....	6
<i>TACT Settings</i> .....	6
<i>Trust anchor store</i> .....	8
<i>Certificate policies database</i> .....	8
<i>Application preferences</i> .....	8
TACT ROLES .....	8
<b>TA STORE MANAGER .....</b>	<b>10</b>
MENUS .....	10
<i>File</i> .....	10
<i>Options</i> .....	11
<i>Tools</i> .....	12
BUTTONS .....	14
<i>Add new TA</i> .....	14
<i>Remove selected TA</i> .....	15
<i>Edit selected TA</i> .....	15
<i>Apply changes</i> .....	16
<i>Discard changes</i> .....	16
TAMP MESSAGE AUTHORIZATION .....	16
TASM USAGE .....	17
<i>TASM usage by TA store managers</i> .....	17
<i>TASM usage by TACT administrators and TACT operators</i> .....	18
<b>TACT SERVER CONFIGURATION AND TACT OPERATOR UTILITIES .....</b>	<b>19</b>
MENUS .....	21
<i>Options</i> .....	21
<i>Tools</i> .....	22
<b>TACT COMPLIANCE ASSESSMENT UTILITY .....</b>	<b>24</b>
MENUS .....	25
<i>Options</i> .....	25
<i>Tools</i> .....	25
<b>TACT COMMAND LINE UTILITY .....</b>	<b>27</b>
COMPARE ACTION .....	27
HASH ACTION .....	28
NEW ACTION .....	28

STATUS ACTION .....	28
UPDATE ACTION .....	28
<b>COMMON TASKS .....</b>	<b>29</b>
EDIT PATH SETTINGS .....	29
<i>Certificate Policies</i> .....	30
<i>Names</i> .....	33
<i>Country Code Filter</i> .....	35
<i>Other Path Settings</i> .....	37
<i>Dynamic path building and revocation status determination settings</i> .....	38
EDIT SECURITY ENVIRONMENTS .....	46
EDIT CERTIFICATE POLICIES DATABASES .....	48
EDIT LOGGING CONFIGURATION .....	50
SERVER MANAGEMENT .....	57
<i>System Security Settings</i> .....	57
<i>Using TACT Management Tools</i> .....	58
<i>File size considerations</i> .....	58
<i>Environmental considerations</i> .....	59
<b>APPENDIX A: SUPPORT .....</b>	<b>60</b>
WEB SITE .....	60
TECHNICAL SUPPORT .....	60
<b>APPENDIX B: FILE FORMATS .....</b>	<b>61</b>
TACT SETTINGS .....	62
TACT TRUST ANCHOR STORE .....	64
CERTIFICATE POLICIES .....	65
PATH SETTINGS .....	66
SECURITY ENVIRONMENT .....	66
TACT RESOURCE .....	66
LOGGING CONFIGURATION .....	66
PKI LOG .....	66
URI LAST MODIFIED .....	67
CRL INFO .....	68
<b>APPENDIX C: ERROR CODES .....</b>	<b>68</b>
<b>APPENDIX D: SAMPLE LOG CONFIGURATION .....</b>	<b>70</b>
<b>APPENDIX E: DEFINITIONS .....</b>	<b>72</b>
<b>APPENDIX F: BIBLIOGRAPHY .....</b>	<b>73</b>
<b>APPENDIX G: SAMPLE SCRIPTS .....</b>	<b>74</b>
IMPORTING AND EXPORTING CERTIFICATE POLICY INFORMATION .....	74
DOWNLOADING AND APPLYING A TAMP UPDATE MESSAGE .....	76
RETRIEVING INFORMATION FROM THE PKI LOG .....	77
<b>APPENDIX H: TAMP PROFILE .....</b>	<b>79</b>
<b>APPENDIX I: TAMP USAGE STRATEGIES .....</b>	<b>80</b>
USING TAMP UPDATE MESSAGES WITH INCREMENTAL CHANGES .....	80
<i>How to</i> .....	80

<i>Pros</i> .....	81
<i>Cons</i> .....	81
USING TAMP UPDATE MESSAGES WITH FULL TA STORE CONTENTS .....	82
<i>How to</i> .....	82
<i>Pros</i> .....	83
<i>Cons</i> .....	83
<b>APPENDIX J: TESTING TACT SETTINGS WITH TSC.....</b>	<b>84</b>
<b>APPENDIX K: DEFAULT TACT CONFIGURATION FILES .....</b>	<b>89</b>
<b>APPENDIX L: ACRONYMS .....</b>	<b>91</b>
<b>APPENDIX M: BACKOUT PLAN .....</b>	<b>92</b>
LINUX.....	92
<i>Disabling an Installed Plug-in</i> .....	92
<i>Removing the TACT Plug-in on Linux</i> .....	92
WINDOWS .....	92
<i>Disabling an Installed Plug-in</i> .....	92
<i>Uninstalling TACT</i> .....	93
<b>APPENDIX N: WINDOWS 8 GUI APPEARANCE .....</b>	<b>94</b>
<b>APPENDIX O: REVERTING V1.2 SETTINGS FILES TO V1.1 AND EARLIER .....</b>	<b>96</b>

## Overview

### Background

The Trust Anchor Constraints Tool (TACT) is intended for use with web servers that require client authenticated Transport Layer Security (TLS) sessions. TACT is intended to address problems that arise from operational practices that are employed to work around defects in the implementation of support for mutual authentication in widely used web server and web browser products. The following diagrams illustrate one manifestation of the problem.

Figure 1 shows a web server operated by an entity within the Department of Defense (DOD). The web server services a community that includes users from partner organizations, such as the Department of State (DOS). For sake of simplicity in these diagrams, assume the trust anchor stores used by the web server and DOS user contain a single trust anchor<sup>1</sup>, with the server relying on the DOD Interoperability Root Certification Authority (IRCA) and the DOS user relying on the DOS Root.

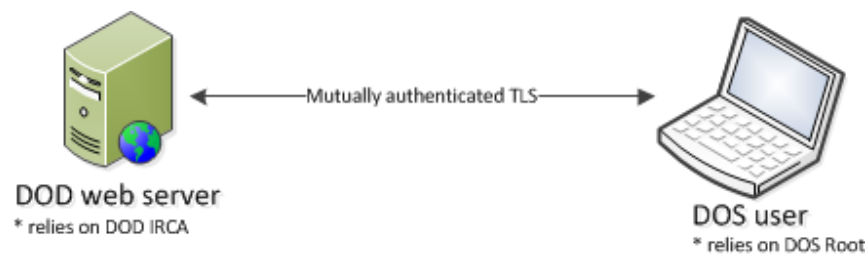


Figure 1 Mutually authenticated TLS example

Certification paths exist from the web server certificate through the Federal Bridge Certification Authority (CA) to the DOS Root and from the DOS user to the DOD IRCA through the Federal Bridge CA to the DOS Root. In other words, it should be possible for these entities to successfully perform mutual authentication. In practice, mutually authenticated TLS is not possible due to the nature of web browser support for client authentication, which does not allow the browser user to select a credential for authentication. A common workaround is illustrated in Figure 2.

---

<sup>1</sup> The relevant point is that the server and the user do not share a common trust anchor.



Figure 2 Mutually authenticated TLS example

In Figure 2, the DOD web server has augmented its trust anchor store to include the DOS Root. This enables mutual authentication to work but it sidesteps the constraints that would have been enforced had the intended certification path from the DOS Root through the Federal Bridge CA to the DOD IRCA been used.

Other manifestations of the problem require actions to be taken on the client machine. For example, when multiple certification paths exist for a client's credential, the browser may elect to use a path that is incompatible with the trust anchors used by the server. Clients typically alter the contents of their local certificate store to work around this problem<sup>2</sup>.

TACT allows server operators to present a larger number of trust anchors to clients, to improve compatibility, without sacrificing constraints on certification path validation.

## Intended Audience

This document is intended for server administrators and Public Key Infrastructure (PKI) administrators working with TACT.

## Supplemental Information

The DoD Public Key Enablement (PKE) web site located at <http://iase.disa.mil/pki-pke> contains many informational documents and best practice guides related to PK-enablement and certificate validation implementation in the DoD. Guidance for the full configuration of Microsoft Internet Information Services (IIS) and Apache web server with both mod\_ssl and mod\_nss is available on the site.

---

<sup>2</sup> See [https://powhatan.iiee.disa.mil/pki-pke/downloads/zip/unclassified\\_fbca\\_crosscert\\_remover\\_v106.zip](https://powhatan.iiee.disa.mil/pki-pke/downloads/zip/unclassified_fbca_crosscert_remover_v106.zip), [http://eca.orc.com/wp-content/uploads/ECA\\_Docs/Diagnose\\_JPAS\\_Trust\\_Issue.pdf](http://eca.orc.com/wp-content/uploads/ECA_Docs/Diagnose_JPAS_Trust_Issue.pdf) or [http://eca.orc.com/wp-content/uploads/ECA\\_Docs/Removing\\_Federal\\_Bridge\\_certificates.pdf](http://eca.orc.com/wp-content/uploads/ECA_Docs/Removing_Federal_Bridge_certificates.pdf) for additional information.

## TACT Overview

### TACT Components

The primary component of TACT is a web server plugin. Versions of the plugin are available for several common web servers including:

- IIS 6.0
- IIS 7.0
- IIS 8.0
- Apache 2.2 with mod\_ssl
- Apache 2.2 with mod\_nss
- Apache 2.4 with mod\_ssl
- Apache 2.4 with mod\_nss

The TACT plugin is supported by a set of management utilities including:

- Trust Anchor (TA) Store Manager (TASM)
- TACT Server Configuration (TSC)
- TACT Operator (TO)
- TACT Compliance Assessment (TCA)
- TACT Command Line (TactCli)

The function of each of these utilities is described in detail in their respective sections of this document.

TACT is also supported by a set of configuration files and databases. The following diagram illustrates the relationship between the TACT components and the various forms of data.



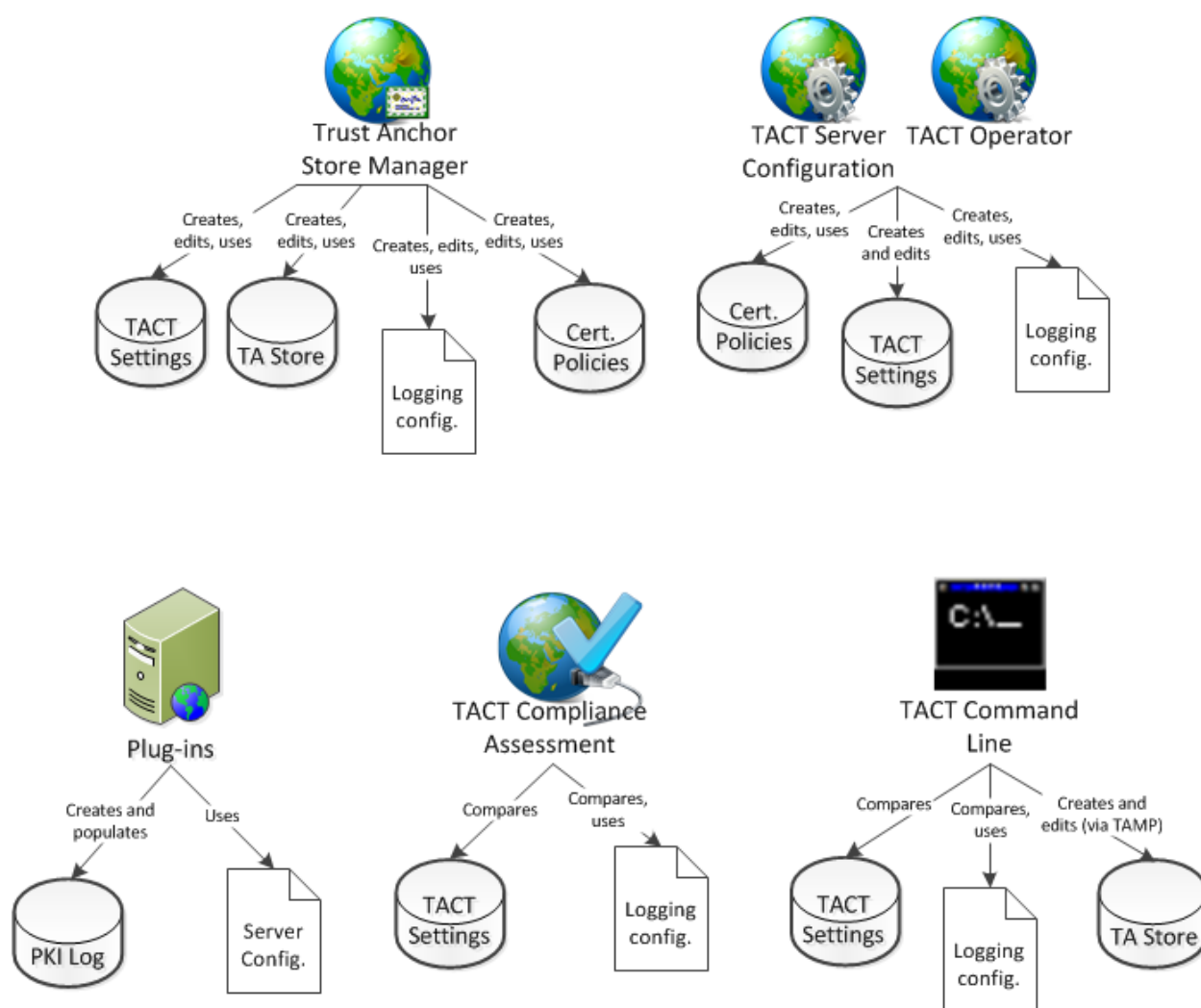


Figure 3 Relationship between TACT components and TACT data

## TACT Data

The TACT plugin relies on several forms of configuration data including certification path validation settings, TACT TA stores, protected resource definitions and other constraints. This configuration data is prepared using the TACT Server Configuration Utility and the TACT Trust Anchor Store Manager Utility<sup>3</sup>. The TACT Compliance Assessment Utility and TactCli utility can be used to confirm a TACT plugin is configured correctly by comparing an authoritative copy of the correct configuration settings with the configuration employed by a TACT plugin. The diagram below illustrates the relationship between these different configuration data.

<sup>3</sup> Web server operators may use TA stores or TA store management messages provided by a TA store administrator instead of the TA Store Manager.

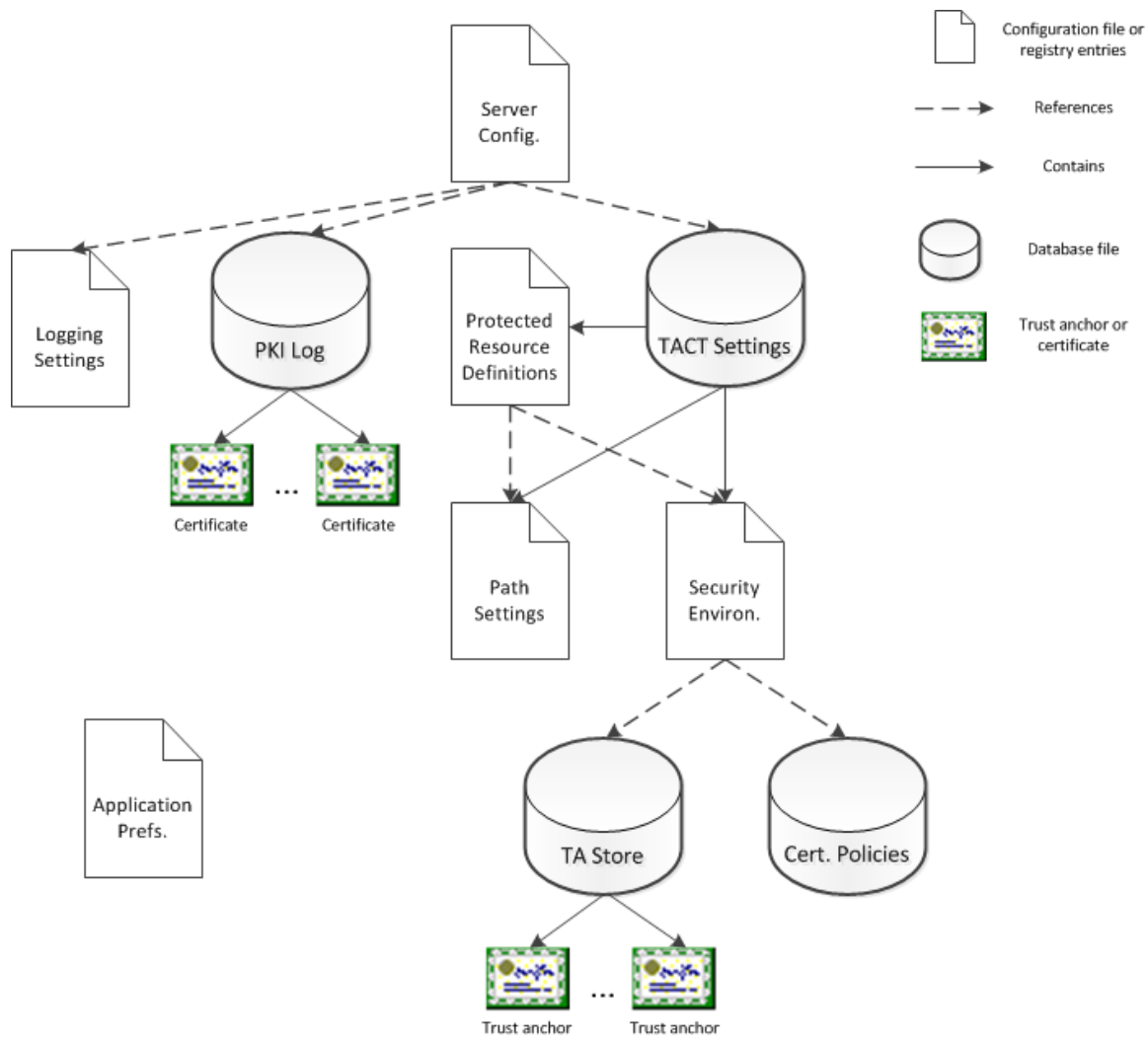


Figure 4 TACT data

The following subsections describe each type of data used by TACT plugins and management applications.

### Server configuration

Server configuration data is used by the TACT plugin to identify the locations of other configuration items, such as trust anchor stores, logging configuration files and TACT settings databases. For Apache servers, this information is included in or referenced by `httpd.conf`. For IIS servers, this information resides in the system registry in the following location:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Red Hound Software\TACT\Plugin`

The TACT Server Configuration Utility can be used to edit IIS settings or the system registry editor. Apache settings are manually edited using a text editor.

### Logging settings

TACT uses the log4cpp library for logging support, which uses a textual configuration file format from the log4j library. An introduction to the logging configuration file format, as well as library capabilities can be found at:

<http://logging.apache.org/log4j/1.2/manual.html>.

TACT provides tools for editing logging configuration files. While these files can be prepared and maintained manually, it is recommended that the provided tools be used to prepare and maintain logging configuration files. The [Edit logging configuration](#) section provides additional information.

### PKI Log

The PKI log is a SQLite database that collects PKI and other session information. There are no configurable aspects of these collection operations. The schema of this database is provided in Appendix B.

### TACT Settings

TACT settings are stored in a SQLite database and include three different types of configuration items: path settings definitions, security environment definitions and protected resource definitions.

#### *Path settings*

Path settings configuration objects contain standard RFC 5280 certification path validation algorithm inputs and other certification path-processing related values. The RFC 5280 configuration items include:

- Initial require explicit policy flag
- Initial inhibit any policy flag
- Initial inhibit policy mapping flag
- Permitted names
- Excluded names
- Initial policy set

Other configuration items include:

- Enforce trust anchor-based constraints (from RFC 5937)
- Enforce cryptographic algorithm and key size constraints (custom flag)

Configuration items related to dynamic certification path discovery include:

- Ignore expired certificates flag
- Retrieve certificates using AIA/SIA extensions – HTTP flag
- Retrieve certificates using AIA/SIA extensions – LDAP flag
- Certificates folder
- Blacklisted certificates

Configuration items related to revocation status determination include:

- Check revocation status flag
- Check OCSP AIA flag
- Check CRLs flag
- Check CRL DP – HTTP flag
- Check CRL DP – LDAP flag
- Only apply CRL grace periods as a last resort flag
- CRL grace period and freshness settings
- Locally trusted OCSP responder settings
- Blacklisted certificates
- Whitelisted (no revocation check) certificates
- CRL folder
- OCSP AIA nonce setting

The [Edit path settings](#) section provides additional information on path settings.

### ***Security environment***

Security environment configuration items are constraints that apply more broadly than path settings configuration items. For example, algorithm and key size constraints are stored in security environment configuration items, as are trust anchor store references and certificate policy database references. These items tend not to vary from one protected resource definition to the next.

The [Edit security environments](#) section provides additional information.

### ***Protected resource***

Protected resource definitions identify a path on the host web server along with a path settings configuration object and a security environment configuration object. The path settings and security configuration objects are evaluated as part of the access control enforcement decision for the resource.

## Trust anchor store

The TACT trust anchor store contains RFC 5914 trust anchors. Each trust anchor is an X.509 certificate wrapped in a TrustAnchorChoice structure, with optional constraints defined. Additional information on the structure of TACT trust anchor stores is provided in [Appendix B](#).

## Certificate policies database

Certificate policies are stored in a SQLite database and used to simplify the creation of certificate policy-related constraints by allowing a web server administrator to define constraints in terms of policies associated with the enterprise of the administrator, i.e., policies that are familiar to the administrator. The database schema is provided in [Appendix B](#). The use of a certificate policies database is optional.

## Application preferences

A variety of application preferences are saved across application invocations. These preferences include window size and placement, last folders used, etc. On Linux, these preferences are saved to a file in the user's home directory. On Windows, these preferences are saved to the system registry under the following registry hive:

- HKEY\_CURRENT\_USER\SOFTWARE\Red Hound Software

Application preferences can be safely deleted. New preferences will be established upon the next execution as necessary.

## TACT Roles

There are three primary roles involved in creating, maintaining and using TACT software and data. These are:

- TACT TA store manager
- TACT administrator
- TACT operator

TACT TA store managers are required to fully understand the PKI architecture used by a TACT installation. TA store managers must have access to trustworthy copies of trust anchors and must understand how to employ RFC 5280 and RFC 5937 constraints to achieve the desired security requirements of the application that uses TACT. TACT TA store managers work primarily with the TA Store Manager utility.

TACT administrators are required to understand how to use TA stores provided by a TA store manager to achieve the security requirements of a TACT installation. This may involve defining certification path validation settings, selecting trust anchor stores

for use and defining cryptographic algorithm or key size constraints. TACT administrators work primarily with the TACT Server Configuration utility.

TACT operators are required to understand which resources on a given web server require protection and must create TACT resource definitions to achieve the security requirements. This involves identifying a resource to be protected and selecting path settings and security environment configurations defined by the TACT administrator. TACT operators work primarily with the TACT Operator utility.

## TA Store Manager

The TA Store Manager (TASM) utility can be used to create and edit TACT trust anchor stores. Editing can be performed through the manual addition, deletion and editing of trust anchor definitions using TASM or by processing a Trust Anchor Management Protocol (TAMP) message generated by a remote trust anchor store manager.

Performing manual editing does not require use of a private key, but preparation of messages for remote consumption does. Correspondingly, processing of remotely prepared TAMP messages requires an authorization check for the signer of the TAMP message. The [TAMP Message Authorization](#) section below provides additional details regarding preparation of TAMP messages for distribution.

## Menus

### File

The file menu can be used to create a new trust anchor store from scratch, create a new trust anchor store from a TAMP status response message, open an existing trust anchor store or save a trust anchor store.

To create a new trust anchor store from scratch, click the **New trust anchor store** menu item, or click Ctrl-N. You will be prompted to select a certificate containing a public key that can be used to validate TAMP messages used to manage the new trust anchor store. The TA store manager for this new trust anchor store must possess the private key corresponding to the selected certificate. If you do not wish to use TAMP to manage the trust anchor store or you do not wish to select a TAMP message signer at this time, click cancel and an empty TA store will be created. A TA store manager can be designated later, if desired. If you do select a TA store manager certificate, the new database will contain the TA store manager public key with TAMP authorization defined.

Add new trust anchors and edit the corresponding constraints using the **Add new TA...** and **Edit selected TA...** buttons. Click **Apply Changes** to generate a TAMP message containing the changes then click the **Save trust anchor store** menu item, or click Ctrl-S, to save the changes. The trust anchor store file is not saved when **Apply Changes** is clicked. You must save the file after click **Apply Changes** prior to closing the file to ensure changes persist.

To discard changes that have not been applied, click the **Discard Changes** button or close the trust anchor store without saving.

To create a new trust anchor store from a TAMP status response message, click the **New trust anchor store from status message** menu item.

To open an existing trust anchor store, click the **Open trust anchor store** menu item, or click Ctrl-O.

## Options

The **Options** menu contains a single item: **Edit options**. The options editor dialog is shown below.

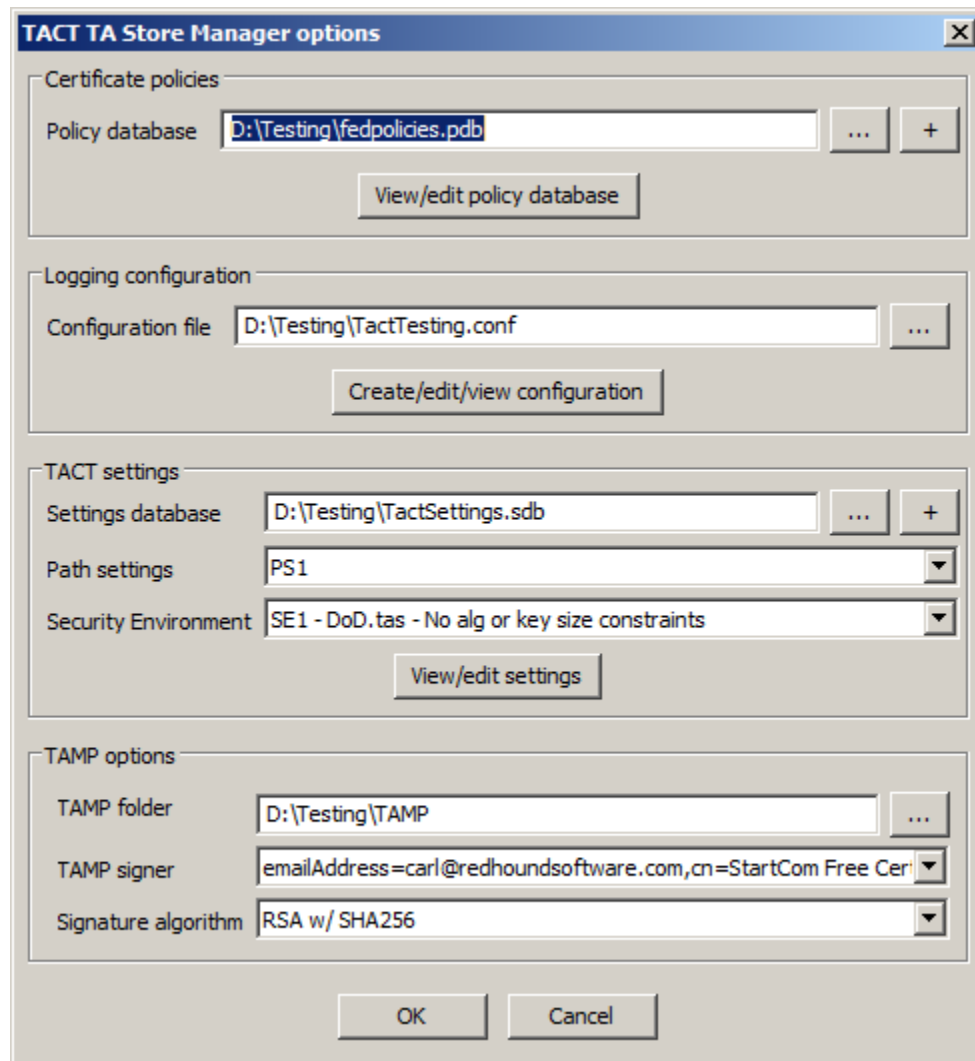


Figure 5 TACT TA Store Manager options editor dialog

The **Certificate policies** box can be used to select, create, view or edit a certificate policies database. The selected database will be used when path settings are edited. Click the ... button to browse to an existing certificate policies database to edit/view. Click the + button to create a new certificate policies database. Click the **View/edit policy database** button to view or edit the contents of the certificate policies database.



In the **Logging configuration** box, click the ... button to browse to a logging configuration file to create or edit/view. Click the **Create/edit/view** button to create a new logging configuration file, if the specified file does not already exist, or to display the contents of the file for viewing and editing, if the specified file does exist.

The **TACT settings** box is used to select a path settings and a security environment setting for use when processing TAMP messages. Click the ... button to select an existing TACT settings database containing the desired settings. To create a new TACT settings database, click the + button then click **View/edit settings** to define at least one path settings configuration item and one security environment configuration item.

The **TAMP options** box is used to establish where TAMP messages should be saved and to define digital signature generation settings. The **TAMP folder** field identifies a folder where TAMP messages are written throughout the usage of TASM. Click the ... button to browse to a folder to receive TAMP messages. TAMP messages are named using the date and time of generation and include an indication of message type and whether or not the message is signed (unsigned messages include .unsigned in the name). The **TAMP signer** list box is used to define the default TAMP message signer. This list is populated with available credentials. When a trust anchor store is opened, if the signer identified here is present in the TA store an authorized for TAMP, the signer is automatically selected for use. Select an empty value to define no default TAMP signer. The **Signature algorithm** field allows for selection of either RSA w/SHA256 or RSA w/SHA1 as the signature algorithm. Note, the TAMP signature generation capability is only present in the Windows editions of TASM in version 1.0. The TAMP Signer drop lists will be empty on other platforms.

## Tools

The **Generate trust anchor status message** can be used to generate and save a TAMP status update message that can be used to create a new database instance containing the contents of the current trust anchor store. When a **TAMP Signer** is selected, the message will be signed, enabling the recipient to verify the signature. Note, this signature is analogous to the signature on a self-signed certificate, i.e., it does not authenticate the source as the signature is verified using a trust anchor within the status message itself. Thus, in addition to verifying the signature, a hash of the file should be checked with the originator of the TAMP Status Response. To securely exchange and use a TAMP Status Response message, perform the following steps:

- Generator steps
  - Select a credential from the **TAMP Signer** drop list (this step is optional)
  - Select the **Generate trust anchor status message** from the **Options** menu to create and save a TAMP status response message.

- Select the **Hash file** menu item from the **Options** menu, browse to the newly generated TAMP status response message, save the message digests for secure transmittal to recipients of the TAMP status response message.
- Recipient steps
  - Authenticate the message contents by selecting the **Hash file** menu item from the **Options** menu, browsing to the received TAMP Status Response message and confirming the generated message digest(s) match the message digest(s) provided by the source of the message. If the message is signed, signature verification will be performed when the message is processed in the next step. To view message signer details, use the **Verify TAMP message signature** menu item after the new trust anchor store has been created.
  - To create a new database using the contents of the message, select the **New trust anchor store from status message** menu item from the **File** menu and browse to the received message. To apply an update message to the open trust anchor store select the **Process trust anchor update message(s)** or **Process folder containing trust anchor update messages** menu items.

The **Process trust anchor update message(s)** menu item can be used to process one or more files containing signed trust anchor update messages. The files are selected using a file browser dialog. This menu item should be used when processing a single trust anchor update message or selected files in a given folder but not all. The **Process folder containing trust anchor update messages** menu item can be used to process one or more files containing signed trust anchor update messages by specifying a folder containing the files to process. All files contained in the folder will be processed (but not file in nested folders, i.e., this is not a recursive tool). This menu item should be used when a folder contains all trust anchor update messages of interest (and no additional files).

The **Verify TAMP message signature** menu item can be used to verify the signature on a TAMP message relative to the open trust anchor store. This would typically be used to view details of the signer of a TAMP update message (signature verification is automatically performed when an externally generated TAMP message is processed). Click the menu item then browse to a signed TAMP message. Results will be displayed in a dialog similar to the figure below.

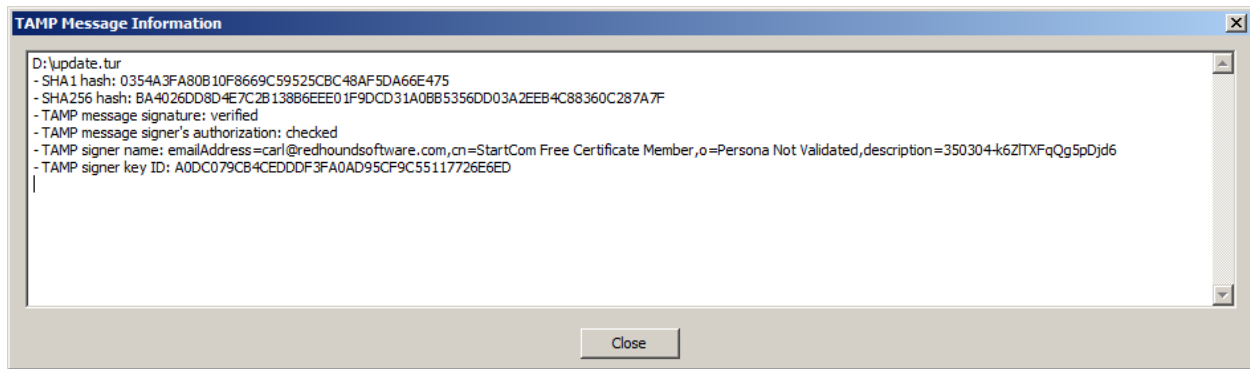


Figure 6 TAMP message information dialog

The **Hash file** menu item can be used to generate and display a SHA1 and SHA256 hash of a selected file. This can be useful to confirm the contents of an unsigned TAMP message, TACT trust anchor store or TACT settings database have not been altered since the file was prepared and hashed by a trusted source of the file. To use, click the Hash file menu item and navigate to the file to hash. A dialog similar to the figure below will be displayed.

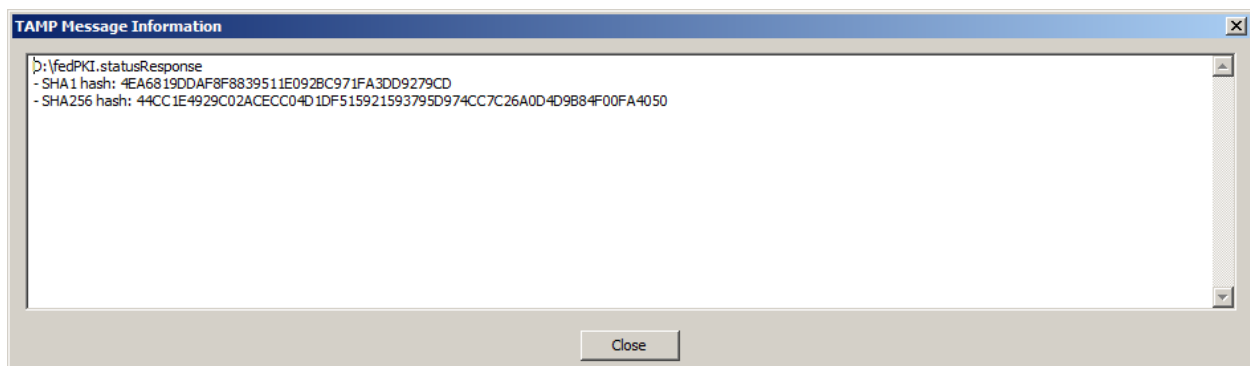


Figure 7 Hash file results dialog

The **Refresh** menu item is not typically used. It can be used to refresh the display to reflect changes applied via an external tool, such as TactCli.

## Buttons

The button below the trust anchors list can be used to add, remove and edit trust anchors. These changes are not performed on demand. Instead, the changes are scheduled and are processed using TAMP messages when the **Apply changes** button is clicked.

### Add new TA

To add a trust anchor to the trust anchor store, click the **Add new TA** button, or click Alt+Shift+A. Navigate to the file containing the trust anchor to add. The file may

contain an X.509 certificate or an RFC 5914 TrustAnchorChoice<sup>4</sup>. The public key in the selected file must not be present in the trust anchor store, else an error message is displayed and the add operation canceled. After selecting a file with a new public key value, the new trust anchor should appear in the list highlighted in green. It can subsequently be removed or edited. The add operation will be completed when the **Apply changes** button is clicked.

### Remove selected TA

To remove a trust anchor from the trust anchor store, select the trust anchor to remove in the list then click the **Remove selected TA** button, or click Alt+Shift+R. The trust anchor will be highlighted in red. The remove operation will be completed when the **Apply changes** buttons is clicked.

### Edit selected TA

To edit a trust anchor, select the trust anchor to remove in the list the click the **Edit selected TA** button, or click Alt+Shift+E. Be aware that constraints applied to a trust anchor effect validation of the entire certification path, not just validation of the end entity certificate. For example, adding a single permitted name constraint to a trust anchor to permit a given end entity will not work if there are intermediate CAs in the path to the end entity. It is highly recommended that the TACT Server Configuration (TSC) certificate validation dialog, described in [Appendix J](#), be used to test settings prior to deployment.

TACT administrators cannot define path settings that exceed the constraints defined in a trust anchor. For example, if a trust anchor definition includes a set of certificate policies, a TACT administrator cannot then define a path settings configuration that allows additional policies. The effective constraints on path validation are essentially as follows (see RFC 5937 for complete rules):

- Intersection of policies from trust anchor and policies from initial policy set
- Intersection of permitted name constraints from trust anchor and initial permitted names (where absence of permitted name constraints in a trust anchor or path settings configuration represents unconstrained).
- Union of excluded name constraints from trust anchor and initial excluded names.
- Flag values are true if true in either the trust anchor or path settings and false otherwise.

---

<sup>4</sup> See [Appendix H](#) for details on the requirements for the TrustAnchorChoice structure when using TACT.

Trust anchor constraints are ignored when the **Enforce trust anchor constraints** flag is unchecked.

### Apply changes

To apply changes defined using the **Add new TA**, **Edit selected TA** and **Remove selected TA** buttons, click the **Apply changes** button. This will cause the generation of a TAMP message containing the changes. The message will be signed using the selected credential, if any. The message will be written to the TAMP folder identified in the TA store manager options. The file name will be the date and time of generation. The message will then be applied to the open trust anchor store with results displayed in a dialog.

### Discard changes

To discard changes defined using the **Add new TA**, **Edit selected TA** and **Remove selected TA** buttons, click the **Discard changes** button (or close the trust anchor store). The trust anchor list will be reverted to the state of the last saved version of the trust anchor file.

## TAMP message authorization

TASM uses TAMP messages for all trust anchor store modifications. For example, when the **Add new TA** button is used to add a new trust anchor to the store a TAMP message is generated and used to perform the update. If the **TAMP Signer** drop list identifies a credential, the TAMP message is signed. If no credential is selected, the TAMP message is not signed. In either case, the TAMP message is written to the TAMP folder identified in the TASM preferences. Unsigned messages are acceptable in this case, because the modifications are being directly performed on a trust anchor store for which the TASM operator has write access.

Modifications may also be performed using the **Tools->Process trust anchor update message** menu item. Only signed messages may be used when performing modifications in this manner. Additionally, the signer of the message must be authorized for TAMP in the trust anchor store being edited. The signature on the message and the authorization of the signer can be checked using the **Tools->Verify TAMP message signature** menu item. A dialog similar to the one shown below is displayed with information about the signer along with message digests calculated over the TAMP message. If signature verification fails or the signer is not authorized, an error dialog is displayed instead.

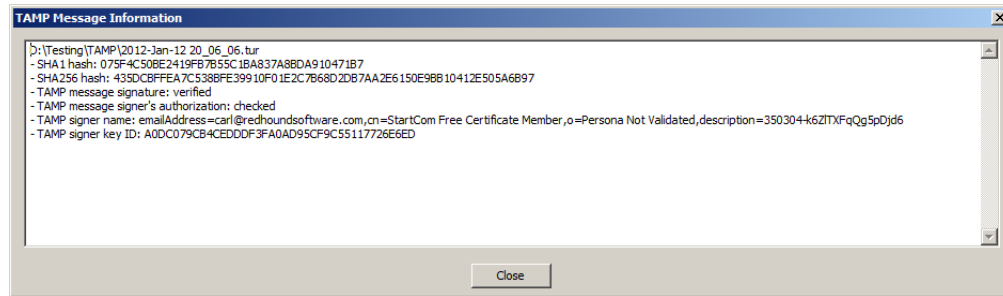


Figure 8 Signature verification results

TAMP authorization can be granted or removed by selecting and right clicking a trust anchor and choosing the **Add TAMP authorization** or **Remove TAMP authorization** item from the context menu, applying the change and saving the trust anchor store. Adding TAMP authorization causes a Cryptographic Message Syntax (CMS) Content Constraints extension to be added to the trust anchor. Removing TAMP authorization removes the extension. The TAMP TA column in TASM indicates which TAs are authorized for TAMP. See [Appendix H](#) for details on TACT's usage of TAMP and associated specifications.

## TASM usage

TASM can be used in various ways, including the following:

- A central management authority may prepare a TA store for use by TACT operators.
- A web server administrator who fulfills all TACT roles may use TASM to prepare a trust anchor store for local use.
- A central management authority may distribute trust anchors using TAMP messages or complete trust anchor stores.
- TACT operators may use TASM to apply TAMP messages received from a TA store manager.

The following sections describe the usage of TASM for trust anchor store managers and by TACT administrators. Usage of TAMP messages for managing trust anchors is described in [Appendix I](#).

### TASM usage by TA store managers

Trust anchor store managers must possess a firm understanding of the PKI environment in which a trust anchor store is to be used. The TASM operator must possess the following materials:

- Trustworthy copies of trust anchors of interest
- Object identifiers associated with certificate policies of interest
- Private key for signing TAMP messages (optional)

- Public key certificates corresponding to private keys that may be used to sign TAMP messages (optional)

The TA store manager will use these materials to prepare TACT trust anchor stores and certificate policies databases for use by TACT plugin installations.

Ideally, the TA store manager will possess or obtain the following materials, which may be made available to TACT administrators for testing:

- Intermediate CA certificates of interest
- End entity certificates of interest

End entity certificates must be obtained from representatives of the communities to be serviced by a TACT installation. Intermediate CA certificates can be similarly obtained or can be obtained using a tool like the [PKI Interoperability Test Tool](#) to harvest certificates from publicly accessible servers.

### **TASM usage by TACT administrators and TACT operators**

TACT administrators and TACT operators are not expected to manually edit trust anchor store contents but may use TASM to apply TAMP messages to a trust anchor store. If more than one TAMP message is to be applied, care must be taken to ensure the messages are applied in sequential (or chronological) order. See [Appendix I](#) for details on trust anchor store management using TAMP messages with TACT management utilities.



## TACT Server Configuration and TACT Operator utilities

TACT administrators use the TACT Server Configuration (TSC) utility to configure protected resources. Each protected resource is associated with a path settings configuration object and a security environment object. In most cases, the TACT administrator will be provided a TACT settings database that has been pre-populated with path settings and security environment objects for use in defining protected resources, or TACT resources. TSC enables the TACT administrators to create, edit, delete, import, export or copy path settings and security environment objects within a TACT settings database. TSC also enables creation of new TACT settings databases. The main TSC interface is shown below. TACT settings databases are portable and can be used on different platforms. However, care must be taken to ensure trust anchor store, certificate policies database and TACT resource paths are adjusted following any migration. Failure to adjust paths can result in denied access.

TACT operators use the TACT Operator utility to configure protected resources. TACT Operator is essentially the TACT Server Configuration utility minus the ability to configure path settings and security environment definitions.

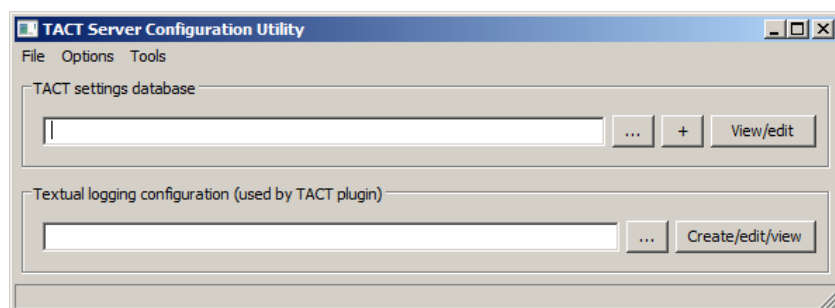


Figure 9 TACT Server Configuration Utility

The **TACT settings database** box features a text box for the full path and filename of a TACT settings database along with three buttons. The ... button can be used to browse to an existing TACT settings database. The + button can be used to create a new TACT settings database. The **View/edit** button can be used to view or edit the contents of the TACT settings database identified in the text control. TACT settings typically reside in C:\TactSettings on Windows and /etc/tact on Linux.

The **Textual logging configuration (used by TACT plugin)** box features a text box for the full path and filename of a logging configuration file along with two buttons. The ... button can be used to browse to an existing logging configuration file or to a location where a new logging configuration file should be created. The **Create/edit/view** button can be used to create a new logging configuration file, if none exists in the specified



location, or to open an existing logging configuration file for viewing or editing. The logging configuration file used by the plugins typically reside in `C:\TactSettings` on Windows and `/etc/tact` on Linux.

TACT settings are edited using the TACT Settings dialog, shown below. This dialog provides access to a collection of path settings definitions, security environment definitions and TACT resource definitions. The dialog features six buttons, each of which can be activated using a shortcut key pressed in tandem with the Alt key. The letter corresponding to each shortcut key appears underlined on each button when the Alt key is pressed.

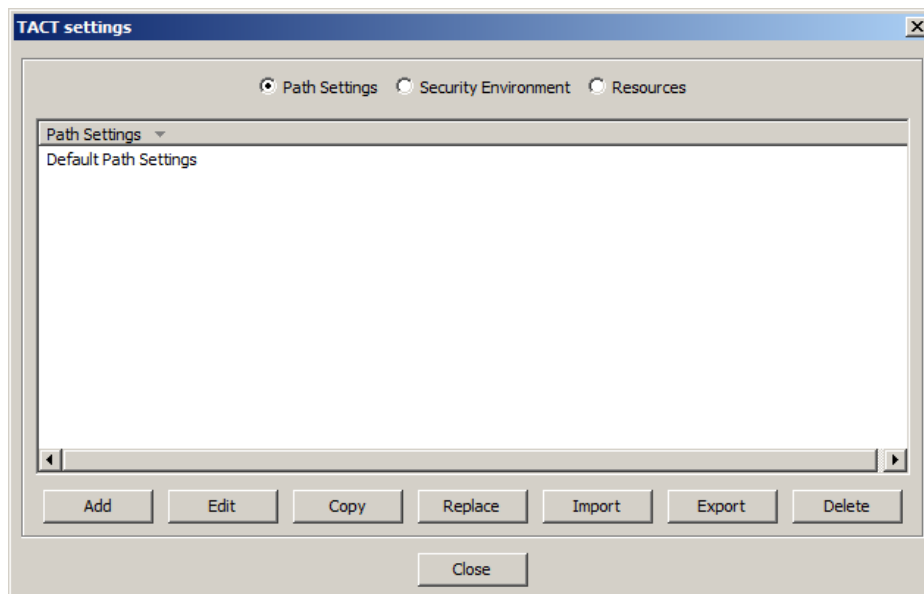


Figure 10 TACT settings editor

The radio buttons can be used to set which type of resource to review or edit. Path settings, security environment definitions and protected resource definitions can be edited using the TACT settings editor. Select the radio button corresponding to the type of item to view or configure. Each path settings item must have a unique name, as must each security environment. Each resource must have a unique path.

The **Add** button can be used to create a new item of the selected type in the TACT settings database. To edit an item, select the item to edit in the list box then click the **Edit** button. [Path settings creation/editing](#) and [security environment creation/editing](#) are covered in the [Common Tasks](#) section below.

Sometimes it is quicker to create a copy of an existing configuration item and make a small number of edits instead of creating a new item from scratch. To create a copy of

an item, select the item to copy in the list box then click the **Copy** button and enter a new name for the item when prompted.

To import an item from a file, click the **Import** button then browse to the file to import.

To export an item, select the item to export in the list box then click the **Export** button and navigate to the location where the file should be saved.

To delete an item, select the item to delete in the list box then click the **Delete** button.

## Menus

### Options

The **Options** menu contains a single item: **Edit TACT Server Configuration Utility options**. The options editor dialog, shown below, enables the selection, creation, viewing or editing of a logging configuration file and certificate policies database.

In the **Logging configuration** box, click the ... button to browse to a logging configuration file to create or edit/view. Click the **Create/edit/view** button to create a new logging configuration file, if the specified file does not already exist, or to display the contents of the file for viewing and editing, if the specified file does exist. On Windows, the logging configuration file is typically located in C:\Program Files\TACT\resources. On Windows, the file is typically located in /etc/tact.

In the **Certificate policies** box, click the ... button to browse to an existing certificate policies database to edit/view. Click the + button to create a new certificate policies database. Click the **View/edit** button to view or edit the contents of the certificate policies database. This database will be used when viewing path settings items.

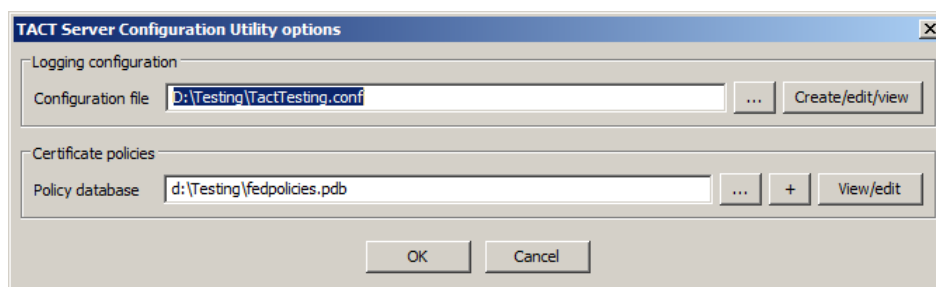


Figure 11 TACT Server Configuration Utility options editor

Logging configuration editing is described in the [Edit logging configuration](#) section below. TSC writes log output to a category named TactServerConfig. To isolate TSC log output from other log output, create a TactServerConfig category and assign it an

appender that receives log information from this category only. Certificate policy editing is described in the [Editing certificate policies databases](#) section below.

## Tools

The **Validate Certificate** menu item can be used to test out path settings and security environment settings against a set of one or more end entity certificates. [Appendix I](#) describes the usage of this tool in detail.

The **Hash file** menu item can be used to generate and display a SHA1 and SHA256 hash of a selected file. This can be useful to confirm the contents of an unsigned TAMP message, TACT trust anchor store or TACT settings database have not been altered since the file was prepared and hashed by a trusted source of the file. To use, click the Hash file menu item and navigate to the file to hash. A dialog similar to the figure below will be displayed.

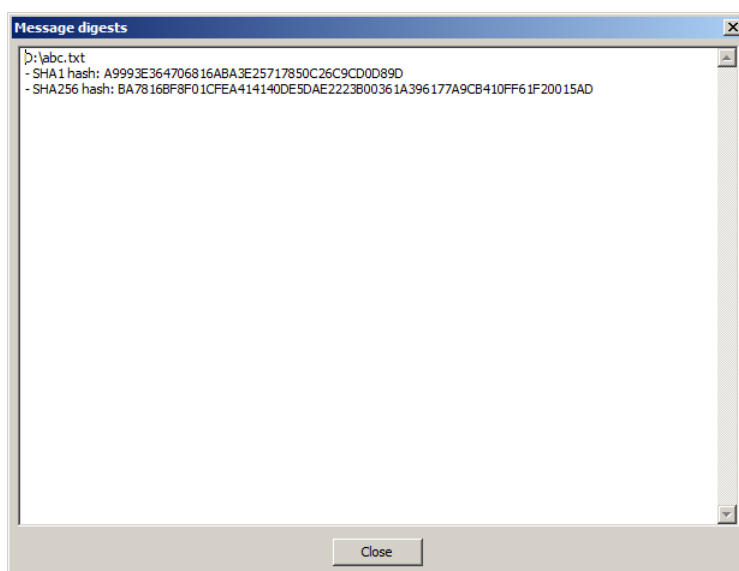


Figure 12 Hash file results

The **Edit TACT registry settings** menu item can be used to alter the locations for information used by IIS TACT plugins. These locations are established when the plugin is installed and can be updated by manually editing the registry or through use of this menu item. The dialog used to edit the registry settings is shown below.

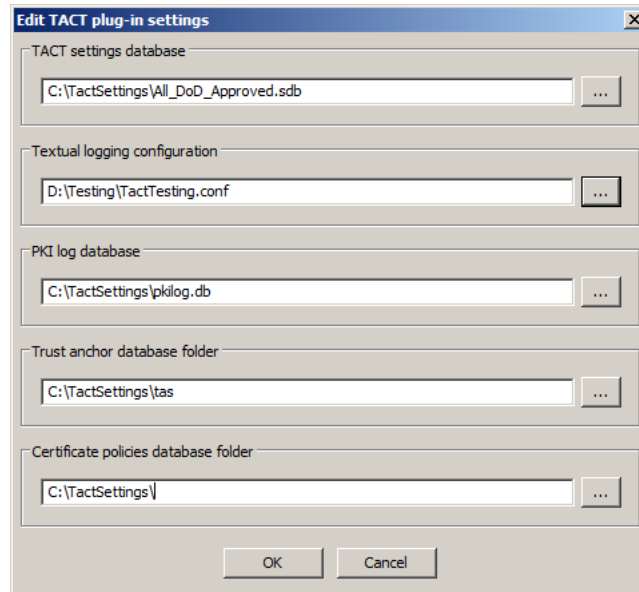


Figure 13 Edit TACT registry settings dialog

## TACT Compliance Assessment Utility

The TACT Compliance Assessment utility can be used to compare the following types of TACT configuration data:

- Trust anchor stores
- Path settings definitions
- Security environment definitions
- Logging configurations
- TACT settings databases

TACT Compliance Assessment (TCA) is useful in environments where central TA store administrators and TACT administrators prepare trust anchor stores and TACT settings for use by TACT operators. The tool enables an auditor to quickly identify areas where configurations are different. The primary TCA interface is shown below. This tool compares settings. It does not analyze settings to determine if one is more restrictive.

Note, comparison operations targeting TACT settings database files do not address the contents of folders identified in TACT settings and do not address the contents of settings database tables introduced in version 1.2.0 (see [file formats](#)).

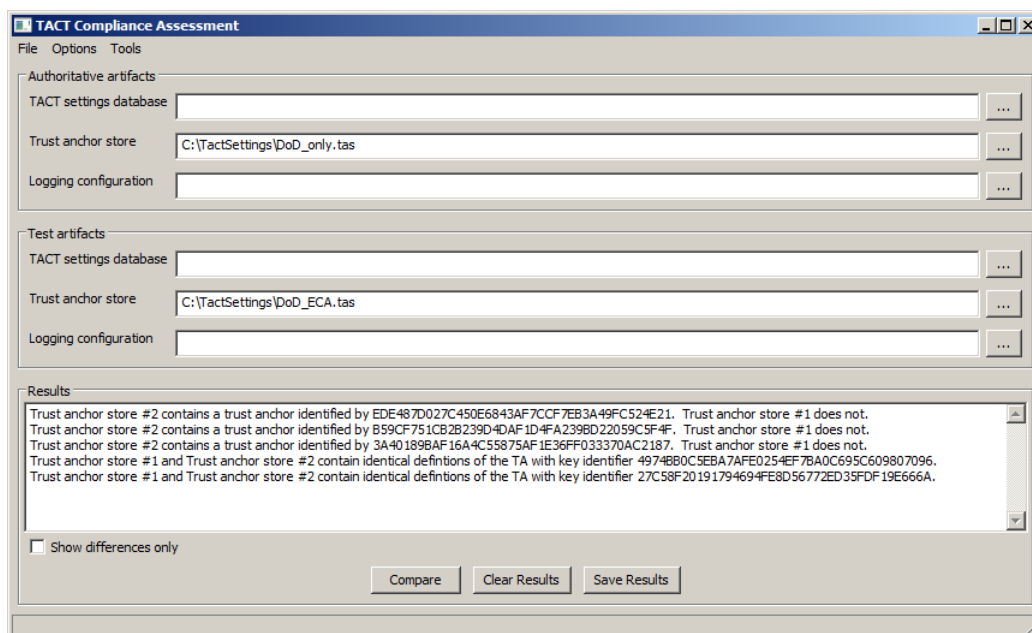


Figure 14 TACT Compliance Assessment

## Menus

### Options

The **Options** menu features a single item: **Edit TACT Compliance Assessment options**. The TCA options dialog is shown below. The dialog enables the selection, creation, viewing or editing of a logging configuration file. Click the ... button to browse to a logging configuration file to create or edit/view. Click the **Create/edit/view** button to create a new logging configuration file, if the specified file does not already exist, or to display the contents of the file for viewing and editing, if the specified file does exist. On Windows, the logging configuration file is typically located in C:\Program Files\TACT\resources. On Windows, the file is typically located in /etc/tact.

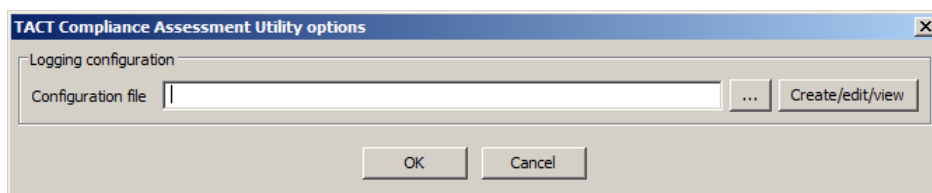


Figure 15 TACT Compliance Assessment utility options editor

Logging configuration editing is described in the [Edit logging configuration](#) section below. TCA writes log output to a category named TactComplianceAssessment. To isolate TCA log output from other log output, create a TactComplianceAssessment category and assign it an appender that receives log information from this category only.

### Tools

The **Tools** menu enables finer grained comparison targets than the primary TCA panel. Individual path settings and security environment definitions saved as files can be compared, in addition to logging configuration files and trust anchor stores. The comparison dialog for each is similar. The figure below shows a path settings comparison dialog as accessed via the **Tools->Compare path settings configurations** menu item.

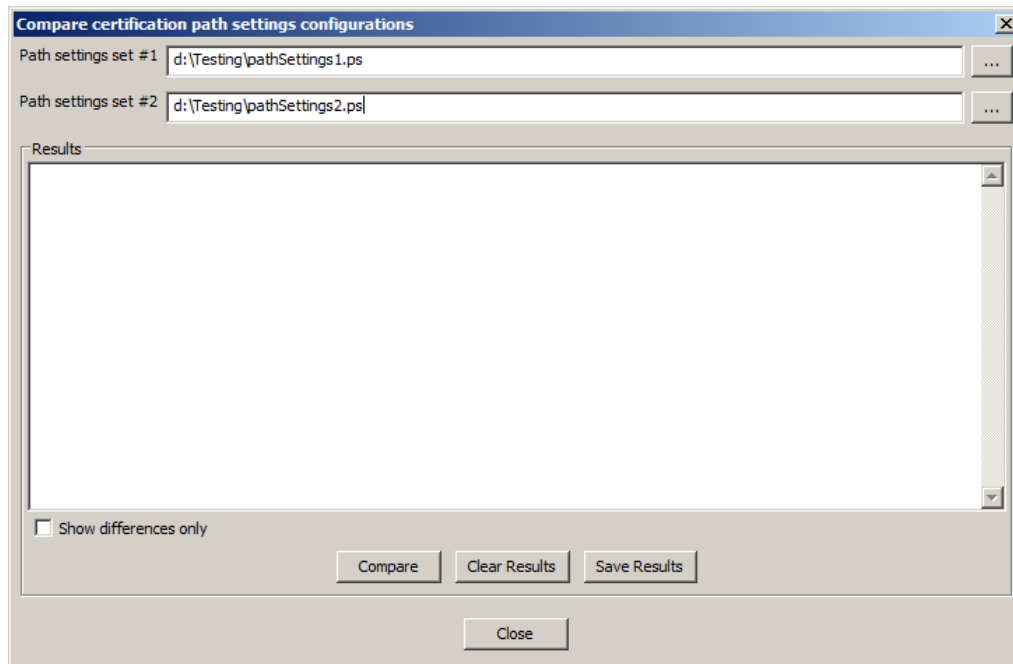


Figure 16 Path settings configuration comparison dialog

Each item in the **Tools** menu can be accessed via a shortcut key, as shown in the table below.

TCA Tool	Shortcut Key
Path settings comparison	F5
Security environment comparison	F6
Trust anchor store comparison	F7
Logging configuration comparison	F8

Figure 17 TCA tools shortcut keys

## TACT Command Line Utility

The TactCli utility is provided primarily to support scripting of some compliance assessment and trust anchor management activities. The usage is as follows:

```
TactCli v1.1.0 (open source release) usage
-h [ --help ]          Print usage instructions
-d [ --diffsOnly ]     Display differences only
-a [ --action ] arg    Action to perform.  Supported actions:
                        compare,hash,new,status,update
--log1 arg             First logging configuration for comparison
--log2 arg             Second logging configuration for comparison
--sdb1 arg             First settings database for comparison
--sdb2 arg             Second settings database for comparison
--tas1 arg             First trust anchor store for comparison
--tas2 arg            Second trust anchor store for comparison
--ps1 arg              First path settings configuration for comparison
--ps2 arg              Second path settings configuration for comparison
--se1 arg              First security environment configuration for
                        comparison
--se2 arg              Second security environment configuration for
                        comparison
--tampUpdate arg       Full path and filename of file containing TAMP update
                        message
--tampUpdateFolder arg Full path of folder containing TAMP update message(s)
--tampConfirm arg      Full path and filename of file to receive TAMP update
                        confirm message
--statusResponse arg  Full path and filename of file containing or to
                        receive TAMP status response message
--database arg         Full path and filename of file of new or existing TACT
                        database
--fileToHash arg       Full path and filename of file to hash
--hashAlg arg          sha1 (default) or sha256
-l [ --logging ] arg   Logging configuration for TactCli
```

TactCli supports the following five actions as described in the following subsections. By default, TactCli displays minimal feedback to the command prompt. This behavior can be altered through the use of the `--logging` parameter to write output to the command line or to another destination. The logging configuration file referenced by the parameter can be prepared using one of the utilities as described in the [Edit logging configuration](#) section.

Though omitted from the examples below for sake of brevity, use of the `--logging` parameter is highly recommended.

### Compare action

The compare action replicates the functionality provided by the [TACT Compliance Assessment utility](#). It can be used to compare path settings, security environments, trust anchor stores, logging configurations and TACT settings databases. To perform a comparison, reference the files to compare with the appropriate pair of parameters. For example, use `--ps1` and `--ps2` to compare path settings files, use `--se1` and `--se2` to



compare security environment files etc. To limit the output to differences only, pass the `-d` parameter. The following example illustrates comparison of a pair of path settings files with only differences reported in the tool output.

```
TactCli --action compare --ps1 d:\ps1.ps --ps2 d:\ps2.ps -d
```

Note, comparison operations do not address the contents of folders identified in TACT settings and do not address the contents of settings database tables introduced in version 1.2.0 (see [file formats](#)).

## Hash action

The hash action generates either a SHA1 or a SHA256 hash of a given file. This can be used to authenticate an unsigned TAMP message, a configuration file, a TACT settings database, etc. The same functionality is also made available via menu items in the [TA Store Manager](#) and [TACT Server Configuration](#) utilities. The following examples illustrate the usage of the hash action. Note, SHA1 is used by default.

```
D:\>TactCli --action hash --fileToHash d:\abc.txt
A9993E364706816ABA3E25717850C26C9CD0D89D
```

```
D:\>TactCli --action hash --fileToHash d:\abc.txt --hashAlg sha256
BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD
```

## New action

The new action can be used to create a new TACT trust anchor store given a TAMP status response message. The trust anchor store can subsequently be managed using the [update](#) action or the [TA Store Manager](#) utility. The following example illustrates the usage of the new action.

```
D:\>TactCli --action new --database d:\newTaStore.tas --statusResponse d:\fedPKI.statusResponse
```

## Status action

The status action can be used to generate a TAMP status response from a given TACT trust anchor store. The following example illustrates the usage of the status action:

```
D:\>TactCli --action status --database d:\newTaStore.tas --statusResponse d:\new.statusResponse
```

## Update action

The update action can be used to present a TAMP update message to a TACT trust anchor store for processing. The following example illustrates the usage of the update action when updating using a single TAMP message file:

```
D:\>TactCli --action update --database d:\newTaStore2.tas --tampUpdate d:\update.tur
```

The following example illustrates the usage of the update action when updating using a folder containing TAMP messages:

```
D:\>TactCli --action update --database d:\newTaStore2.tas --tampUpdateFolder d:\updates
```

## Common Tasks

### Edit path settings

Path settings can be viewed and edited in several contexts in the TACT management applications. When editing a TACT settings database, the path settings editor includes a **Path settings name** field that can be used to change the name of a path settings configuration item. In other contexts, the name cannot be changed and the **Path settings name** field is absent. The figures in this section all include the **Path settings name** field, as shown below.

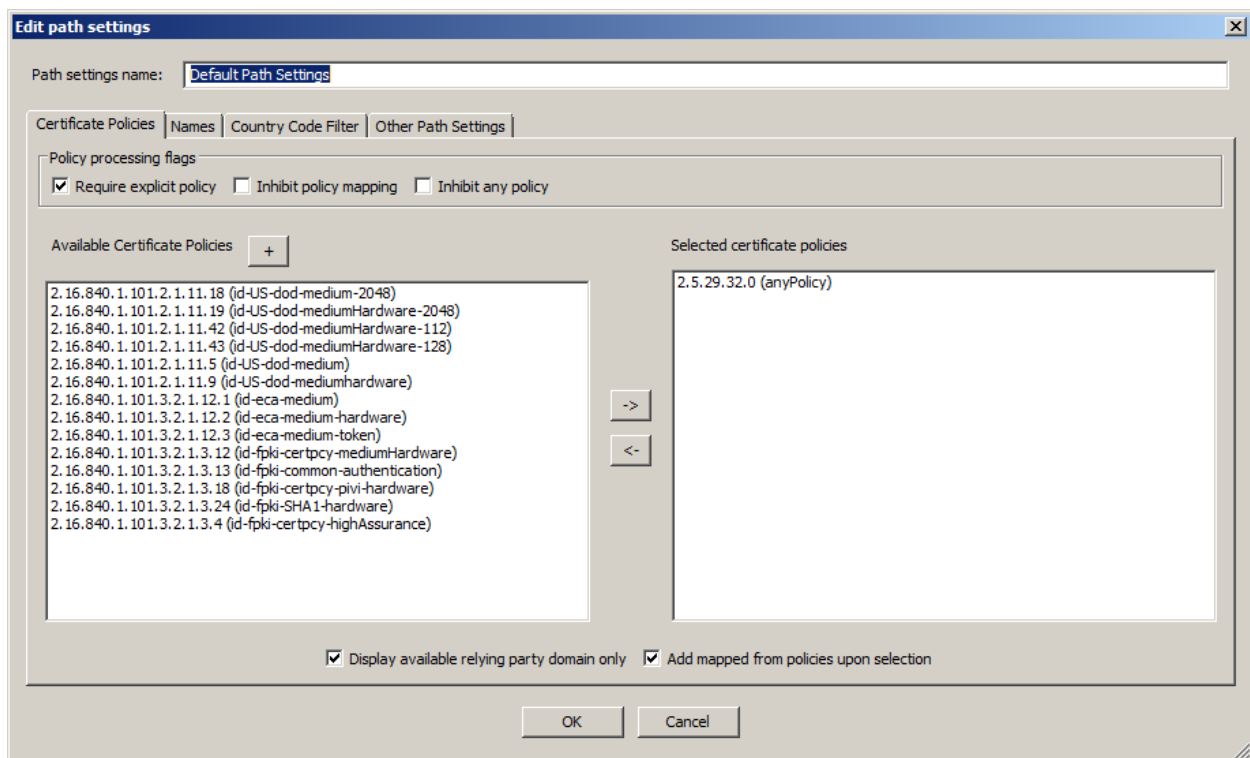


Figure 18 Path settings editor

In addition to the **Path settings name** field, the path settings editor features three tabs: **Certificate Policies**, **Names**, **Other Path Settings**. The contents of each tab are described below.

## Certificate Policies

The **Certificate Policies** tab is used to define four certificate policy-related RFC 5280 certification path validation [inputs](#):

- The **Selected certificate policies** list box contents are used as the user-initial-policy-set input.
- The **Require explicit policy** checkbox is used as the initial-explicit-policy input.
- The **Inhibit policy mapping** checkbox is used as the initial-policy-mapping-inhibit input.
- The **Inhibit any policy** is used as the initial-any-policy-inhibit input.

The **Available Certificate Policies** list is populated with the contents of a TACT [certificate policies database](#). To add a policy to the **Selected certificate policies** list select the policy to add in the **Available Certificate Policies** list box and click the **->** button. To remove a policy from the **Selected certificate policies** list select the policy to remove in the **Selected Certificate Policies** list box and click the **<-** button. To add a policy to the **Available Certificate Policies** list box, click the **+** button and enter the desired information in the resulting dialog.

The **Display available relying party domain only** check box can be used to limit the number of policies displayed in the **Available Certificate Policies** list box. For this to be useful, the TACT certificate policies database must be defined with appropriate certificate policies marked as being in the relying party domain. The policies so marked will vary from one enterprise to another. For example, DOD certificate policies will be marked as relying party domain for usage by a DOD application but not marked as being in the relying party domain for usage by an application operated by a partner enterprise. The figure below shows the policy definition dialog.

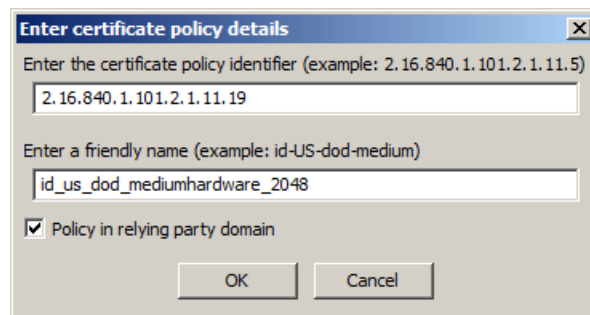


Figure 19 Certificate policy definition

The figure below shows the **Certificate Policies** tab with the **Display available relying party domain only** check box checked while using a TACT certificate policies database that has a set of DOD policies marked as being in the relying party domain.

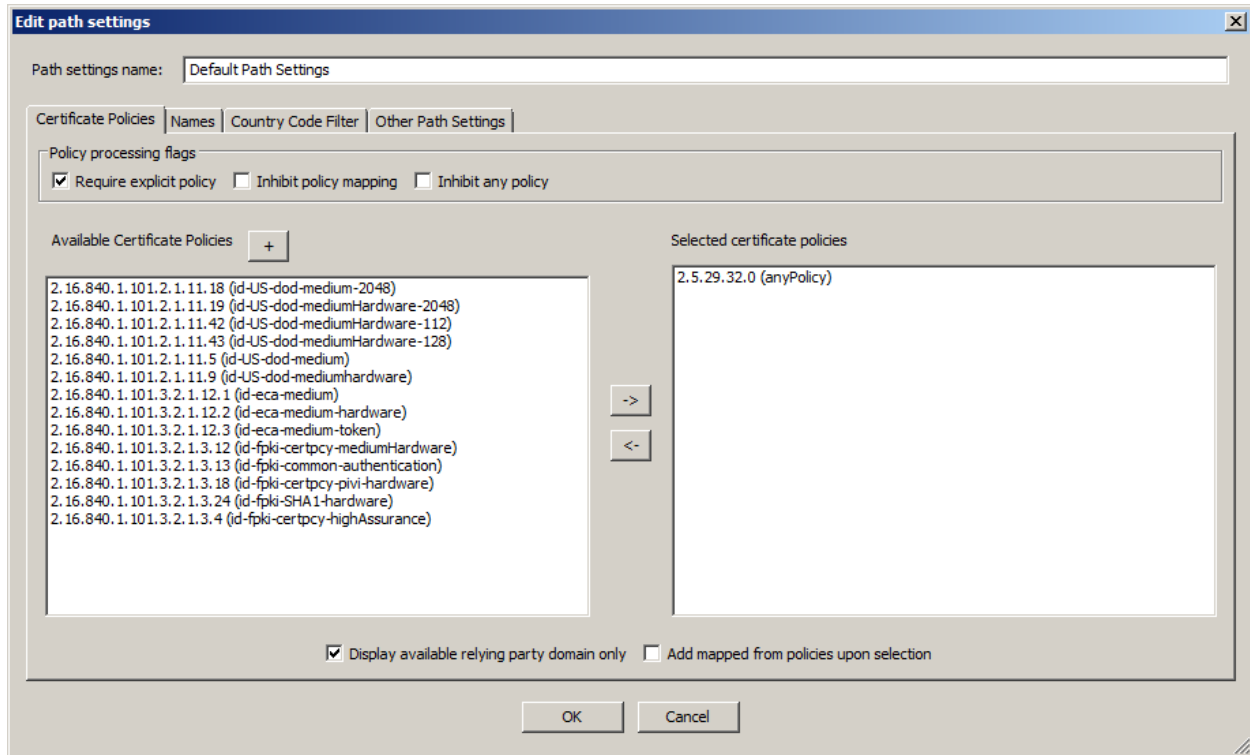


Figure 20 Relying party policies only displayed

The **Add mapped from policies upon selection** can be used to move a set of policies from the **Available Certificate Policies** list to the **Selected Certificate Policies** list. When this option is checked, any certificate policies that have been mapped to a policy selected in the **Available Certificate Policies** list will be moved to the **Selected Certificate Policies** list when the **->** button is clicked. The figure below shows the Certificate Policy Editor dialog with a DOD policy selected that has one associated mapping.

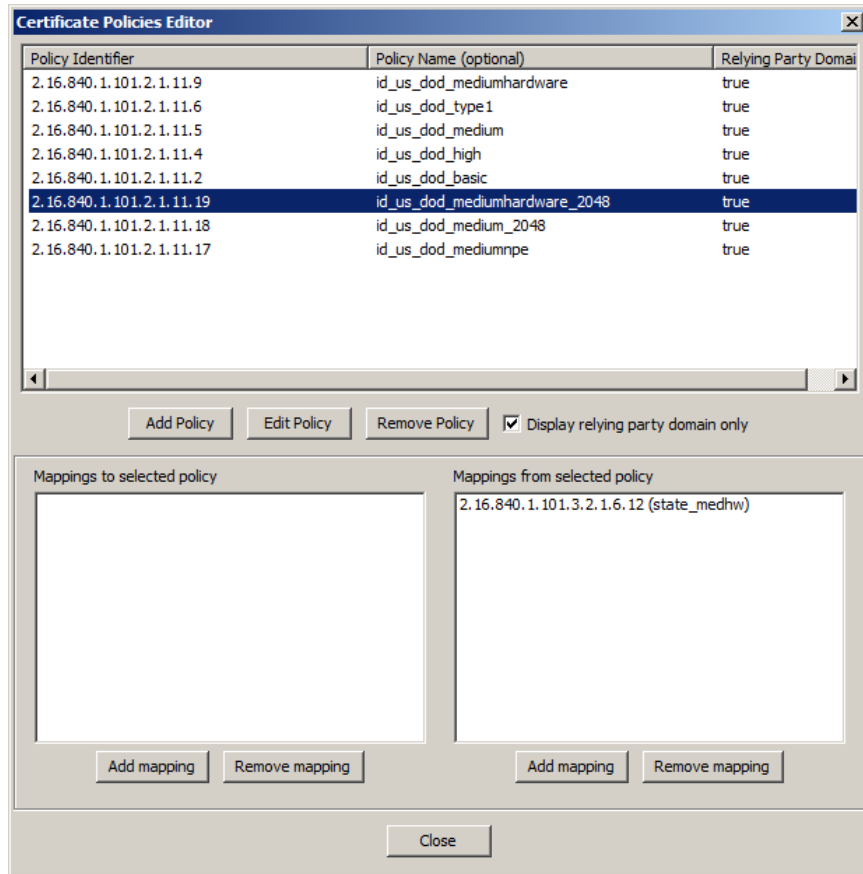


Figure 21 Example policy definition with associated mapping

The figure below shows the result of moving this policy when the **Add mapped to policies upon selection** option is checked. Note, the **Selected certificate policies** list always shows a complete list of policies that have been selected. Only the display of the **Available certificate policies** list can be limited using the **Display available relying party domain only** check box.

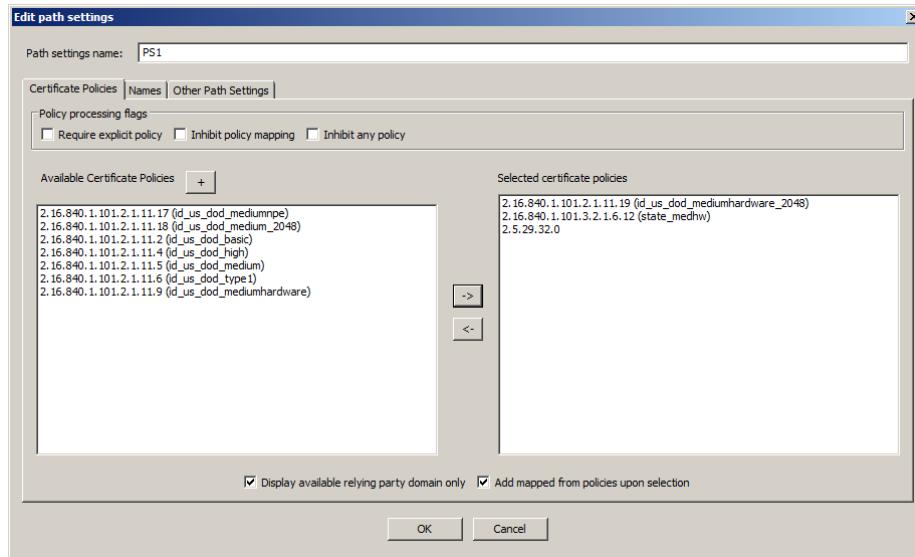


Figure 22 Auto-selection of mapped policies

## Names

Permitted and excluded name constraints can be defined on the **Names** tab, shown below.

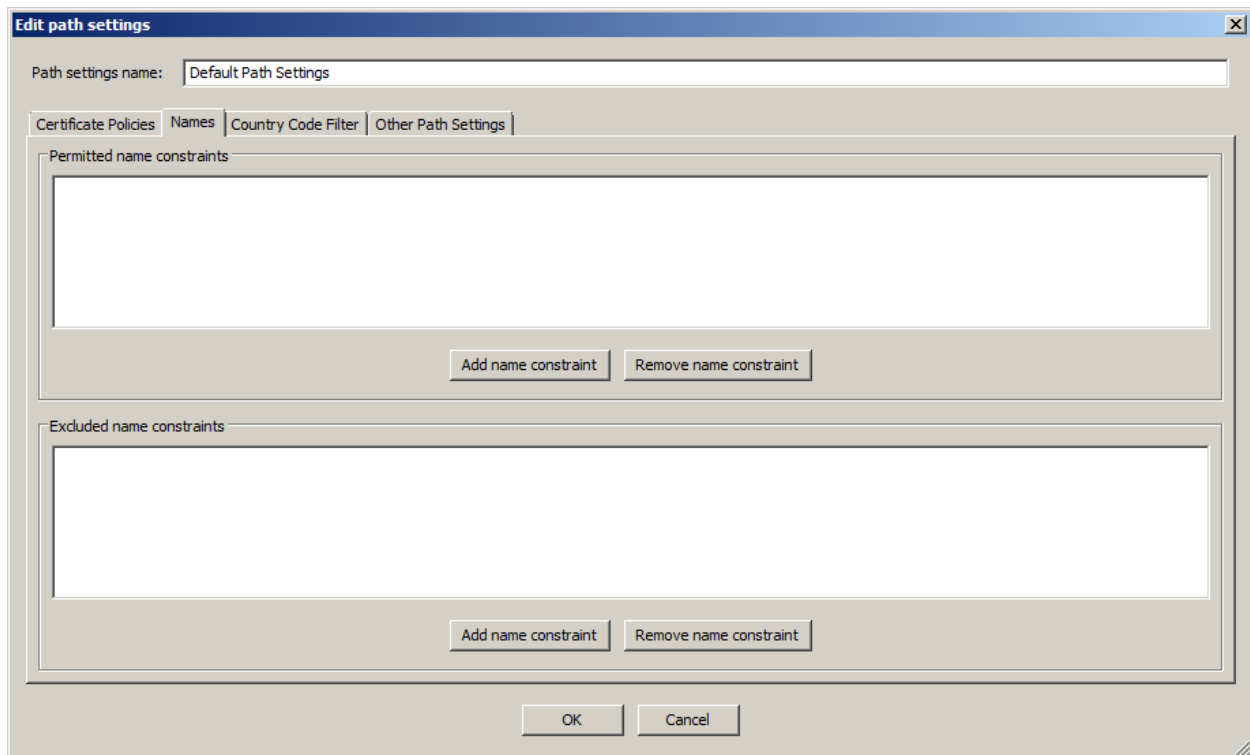


Figure 23 Names tab

To add a permitted name, click the **Add name constraint** button in the **Permitted name constraints** box. A dialog like the one shown below will be displayed. This dialog

allows specification of the following name forms: Distinguished names, URIs, DNS names, RFC 822 names and UPNs.

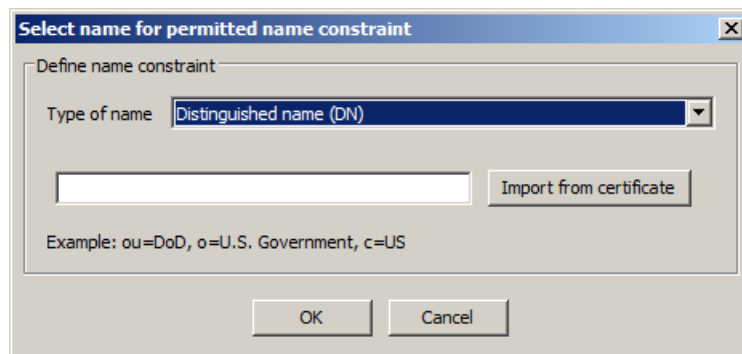


Figure 24 Name constraint entry dialog

To import a distinguished name, click the **Import from certificate** button, browse to a certificate and click on the desired Relative Distinguished Name (RDN). The dialog below shows selections of the following namespace as imported from the DOD Root CA2 certificate: ou=DoD, o=U.S. Government, c=US.

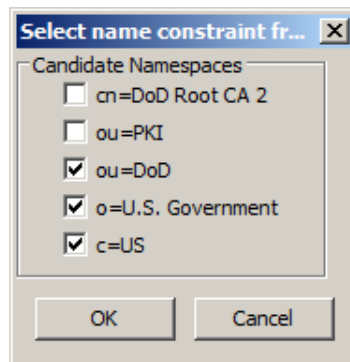


Figure 25 Importing a distinguished name

All other name forms are simply entered into the text box. The figure below shows specification of an RFC 822 name form.

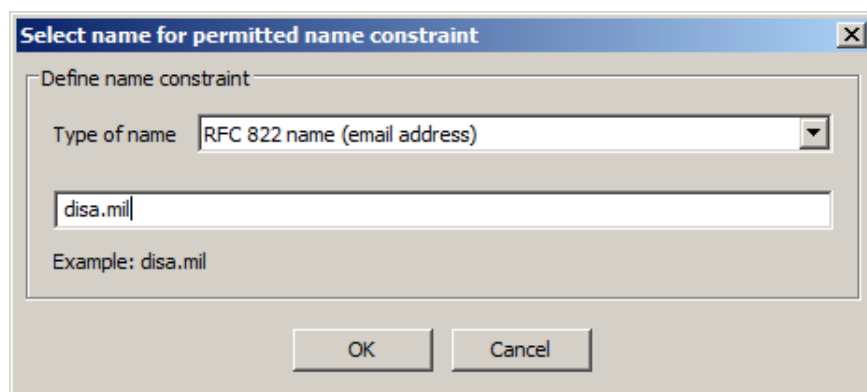


Figure 26 Specifying an RFC 822 name constraint

Use the **Type of name** drop list to select alternative name forms. To remove a previously specified name, select the name to remove then click the **Remove name constraint** button.

When defining name constraints, it is important to keep in mind that the constraints will be processed per RFC 5280 and RFC 5937 rules. This can result in some relatively counterintuitive conditions. For example, a specific individual can be denied access using a single excluded name constraint but a specific individual cannot be granted accessing using a single permitted name constraint. This is due to the nature of path processing which would result in an intermediate CA certificate failing to satisfy the permitted name constraint. Similarly, name constraints only apply to certificates that include a given name form. Thus, it is not possible to permit, for example, all mil UPNs because this would also allow certificates that do not contain a UPN.

### Country Code Filter

The **Country Code Filter** tab, shown below, features three list boxes that allow for specification of sets of permitted or excluded country codes and a check box that can be used to require the presence of a country code value in end entity certificates in order for the certificate to be successfully validated.



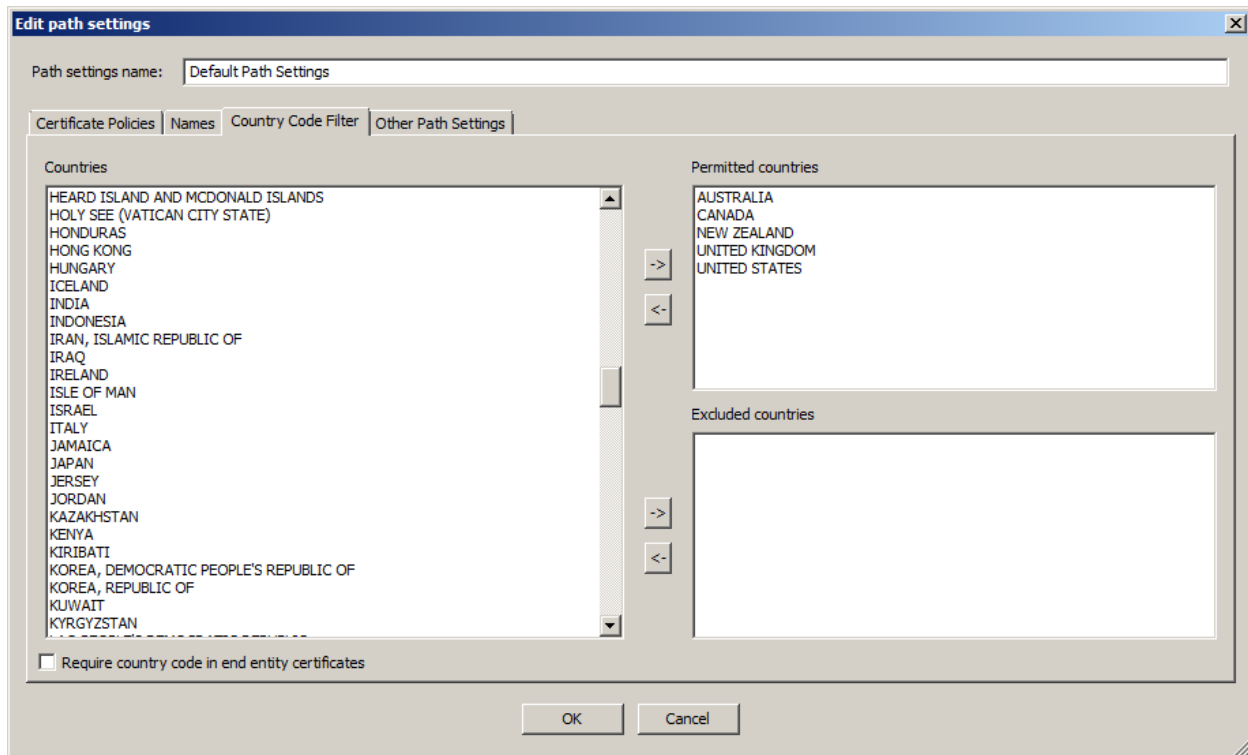


Figure 27 Country code filter dialog

The **Countries** list box features a list of countries from ISO 3166:

[http://www.iso.org/iso/country\\_codes.htm](http://www.iso.org/iso/country_codes.htm). One or more countries can be selected to move into either the Permitted Countries or Excluded Countries list boxes using the appropriate <- or -> buttons. The **Require country code in end entity certificates** check box can be used to require all end entity certificates to include a country code in order to be successfully validated.

During certification path validation, when the **Require country code in end entity certificates** box is not checked, end entity certificates that do not contain a country code value in the subjectDirectoryAttributes extension pass the country code filter check. When the **Require country code in end entity certificates** box is checked, end entity certificates that do not contain a country code value in the subjectDirectoryAttributes extension fail the country code filter check.

When the **Excluded countries** list is not empty, the contents of the country code value in the subjectDirectoryAttributes extension in end entity certificates are evaluated. Certificates containing a value that matches an item in the **Excluded countries** list fail the country code filter check.

When the **Permitted countries** list is not empty, the contents of the country code value in the subjectDirectoryAttributes extension in end entity certificates are evaluated.

Certificates that do not contain at least one value that matches an item in the **Permitted countries** list fail the country code filter check.

## Other Path Settings

The **Other Path Settings** tab, shown below, features two check boxes that allow processing of information from an associated Security Environment to be optionally enabled or disabled.

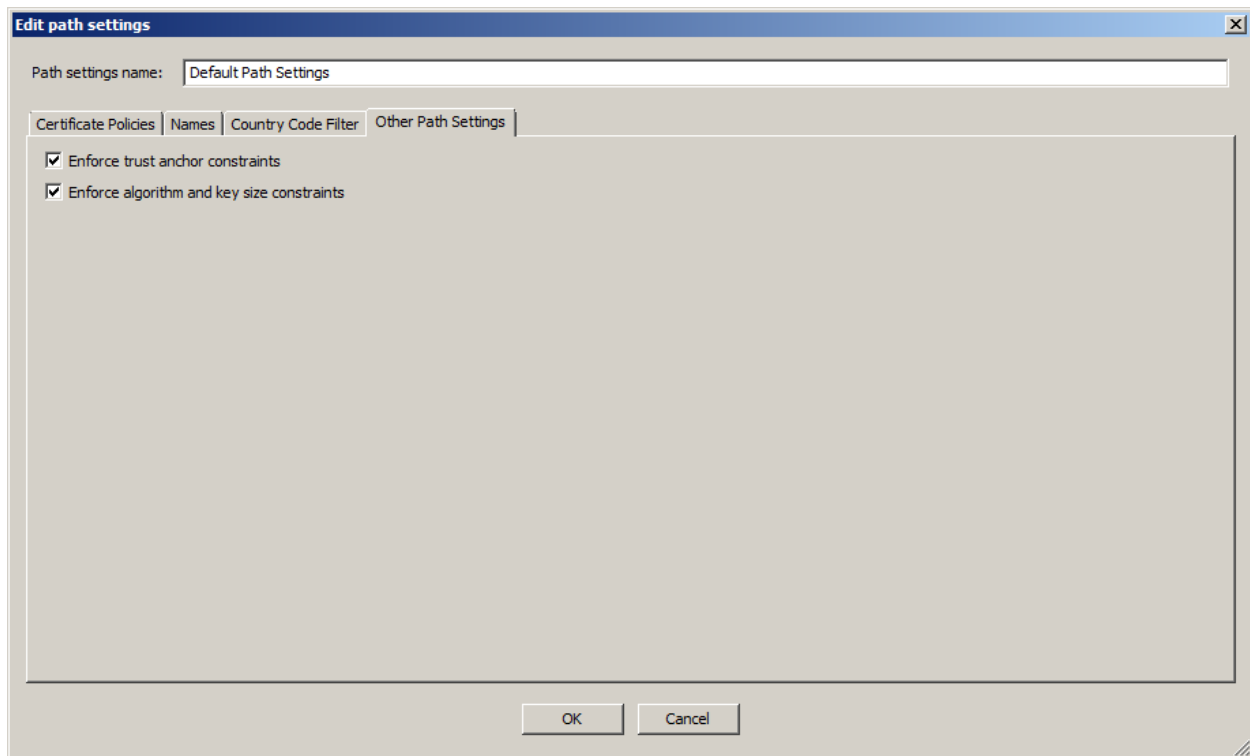


Figure 28 Other Path Settings panel

The **Enforce trust anchor constraints** check box corresponds to the `enforceTrustAnchorConstraints` certificate path validation input defined in RFC 5937. When this box is checked, trust anchor-based constraints are enforced per RFC 5937. When it is not checked, trust anchor-based constraints are not enforced. The **Enforce algorithm and key size constraints** check box enables enforcement of cryptographic algorithm and key size constraints defined in the associated security environment. When this box is checked, algorithm and key size constraints are enforced. When this box is not checked, algorithm and key size constraints are not enforced.

## Dynamic path building and revocation status determination settings

These settings apply only to TACT v1.2 and later and only to TACT as integrated with mod\_nss. The interfaces to edit the settings are present on all platforms, but the settings are only exercised within TACT when integrated with mod\_nss<sup>5</sup>.

### *Path Building Settings*

The **Path Building Settings** tab defines the basic rules for building certificate graphs and obtaining certificates from remote sources. TACT can obtain certificates from a local file folder, from remote servers accessed via HTTP or from remote servers accessed via LDAP. The **Retrieve certificates using AIA/SIA extensions - HTTP** and **Retrieve certificates using AIA/SIA extensions - LDAP** settings determine if remote sources are used. If both are unchecked, no remote sources are consulted to obtain additional certificates. The **Certificates folder** setting identifies the local file folder containing certificates to use when preparing a certificate graph for use during certification path discovery. The folder will be recursively searched for .der, .cer, .crt and .pem files. The **Ignore expired certificates** setting determines whether expired certificates are included in a graph used for certification path discovery. When this option is checked, expired CA certificates are not used. The **Blacklisted certificates** box identifies any additional certificates that should be excluded from consideration during certification path discovery. Click the **Add certificate** button to browse to a certificate file to add to this list or select a certificate in the list and click the **Remove selected certificates** button to remove a certificate from the list.

---

<sup>5</sup> The settings may also be used by other utilities, like the PKI Interoperability Test Tool version 2 (PITv2).

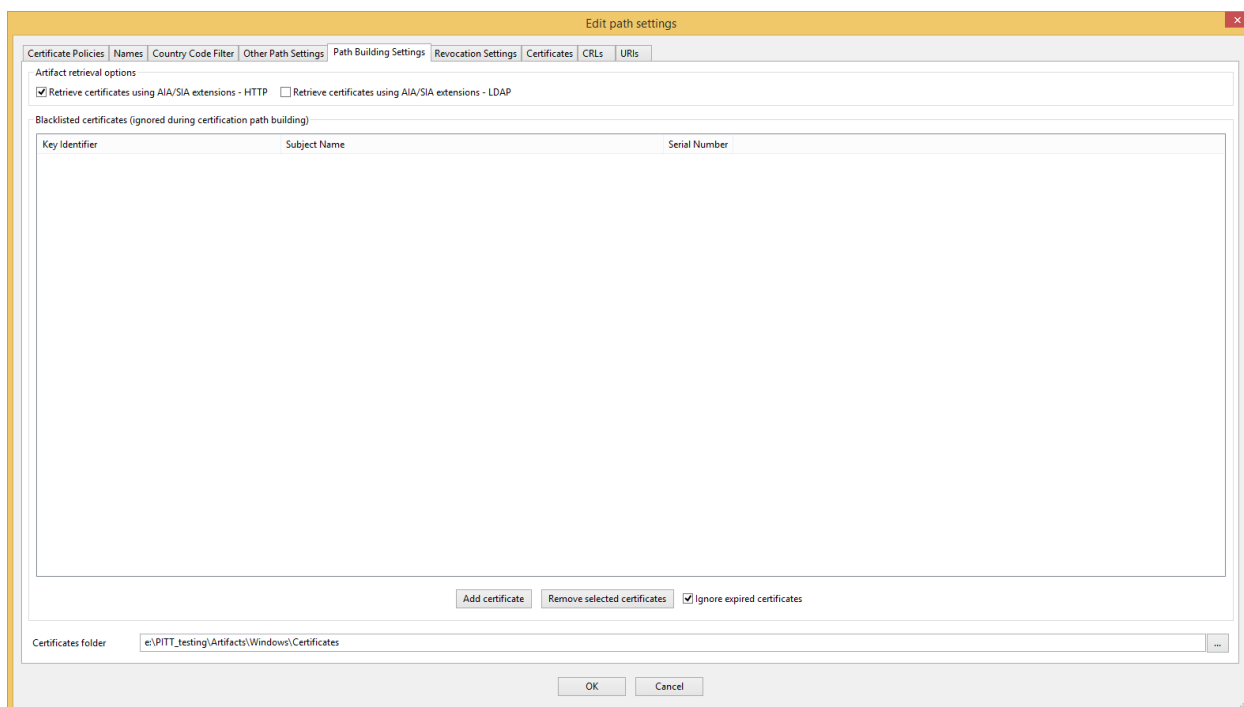


Figure 29 Path building settings

### Revocation Settings

The **Revocation Settings** tab is used to configure the mechanisms used to determine the revocation status of certificates and how CRL time values are processed. The **Check revocation status** setting determines whether revocation status determination checks will be performed at all. When this is checked, revocation status will be checked in accord with other settings. When this is not checked, no attempt will be made to determine the revocation status of any certificate. The check boxes below the **Mechanisms** list control the commonly used options.

When the **Check OCSP AIA** setting is checked, OCSP responders identified in the authorityInformationAccess (AIA) extension of a certificate being validated may be consulted for revocation status information. When processing an OCSP response obtained via a URI from an AIA extension the response must be verified using the CA certificate used to verify the certificate containing the AIA extension or the responder's certificate must be verified using the CA certificate used to verify the certificate containing the AIA extension. Where the responder is not the CA, the responder's certificate must be included in the response. The **OCSP AIA Nonce** setting determines how nonces are used (or not used) in requests sent to responder identified in an AIA extension.

When the **Check CRLs** setting is checked, CRLs may be processed to determine revocation status information. If a **CRL folder** is specified, locally available CRLs will always be consulted before OCSP or remote CRL locations. Locations identified in a CRL distribution points (CRL DPs) extension will be consulted if the **Check CRL DP - HTTP** or **Check CRL DP - LDAP** settings are checked, with HTTP sources being consulted first if both options are checked. If a revocation freshness configuration is defined to allow stale CRLs to be used and the **Only apply CRL grace periods as a last resort** is checked, then an attempt to retrieve fresh information will be made before relying on stale information.

The screenshot shows the 'Edit path settings' dialog box with the 'Revocation Settings' tab selected. The 'Check revocation status' checkbox is checked. Below it is the 'Revocation freshness configuration' section with a table that has columns for Name, Mechanism, Grace Period, Freshness, and Scope. Below the table are buttons for 'Add configuration', 'Edit configuration', 'Remove configuration', 'Move up', and 'Move down'. A note states: 'By default, stale revocation information is not accepted and fresh revocation information is not required.' Below this is the 'Mechanisms' section with a 'Type' dropdown set to 'Locally trusted OCSP'. Under 'Locally trusted OCSP responder' is a table with columns for Name, URI, Scope, Authorization, and Nonce, with buttons for 'Add responder', 'Edit responder entry', 'Remove responder', 'Move up', and 'Move down'. At the bottom are checkboxes for 'Check OCSP AIA', 'Check CRLs', 'Check CRL DP - HTTP', 'Check CRL DP - LDAP', and 'Only apply CRL grace periods as a last resort'. There is also a 'CRL folder' text box containing 'E:\PITT\_testing\Artifacts\VeryOldCrls' and an 'OCSP AIA Nonce' dropdown set to 'Do not send nonce'. 'OK' and 'Cancel' buttons are at the bottom.

Figure 30 Revocation settings

To add a revocation freshness configuration click the **Add configuration** button below the **Revocation freshness configuration** list. A dialog like the one below will be displayed. Enter a configuration name and select a mechanism (only CRLs are supported in this release). Enter a **Grace period in hours** to allow use of stale CRLs. The grace period defines the maximum allowable difference between the current time and the nextUpdate value in a CRL. Enter a **Freshness in hours** to disallow revocation information that was generated too far in the past (even if not stale). The freshness value defines the maximum allowable difference between the current time and the thisUpdate value in a CRL. The scope of the revocation freshness configuration can be set to all CRLs or limited by CA certificate. For example, to allow stale CRLs only for a given CA set the configurations as described above then click the **Certificate used to validate artifact** radio button and browse to the CA certificate.

Figure 31 Revocation freshness configuration

In addition to the mechanisms described above (i.e., **CRL folder**, **Check OCSP AIA** and etc.), three additional mechanisms may be defined: **Locally trusted OCSP**, **Blacklist** and **No check**. The **Blacklist** mechanism simply lists certificates that are marked as revoked independent of and without consulting any OCSP or CRL sources. The **No check** option is essentially a whitelist. Certificates included listed in the **No check** table are marked as not revoked independent of and without consulting any OCSP or CRL sources.

To configure a locally trusted OCSP responder, choose **Locally trusted OCSP** option in the **Type** field in the **Mechanisms** box then click the **Add responder** button. A dialog box similar to the dialog shown below will be displayed. Enter a name for the configuration in the **Config name** field and enter the URI where the OCSP responder can be reached in the **URI** field then select a **Nonce setting**. When **Do not send nonce** is selected, requests sent to this responder per this configuration will not include a nonce. When either **Send nonce and require match in response** or **Send nonce but do not require match in response** is selected, a nonce will be included in requests with response processing proceeding per the selected option. The **Send nonce but do not require match in response** option is primarily useful when the responder does or may return cached responses. If a freshly generated response is required, use the **Send nonce and require match in response** option, keeping in mind that the responder may not comply and responses that do not match will be discarded and not be used for revocation status determination.

The **Scope** option can be used to limit the applicability of the configuration. When **All certificates** is selected, the configuration may be used to determine the revocation status of any certificate. This is not typically used. The **Subject namespace** and **Issuer namespace** options can be used to limit applicability to certificates with a subject or issuer name subordinate to the configured name. The **Verified using key** option limits the applicability to certificates verified using the public key in the specified certificate.

The **Authorization** setting determines how the responder is authorized. When the **Responses verified using key** option is used, responses retrieved per this configuration must be verified using the public key in the specified certificate. When the **Responder certificate verified using key** option is selected, the response must be verified using a responder certificate that can be verified using the public key in the specified certificate.

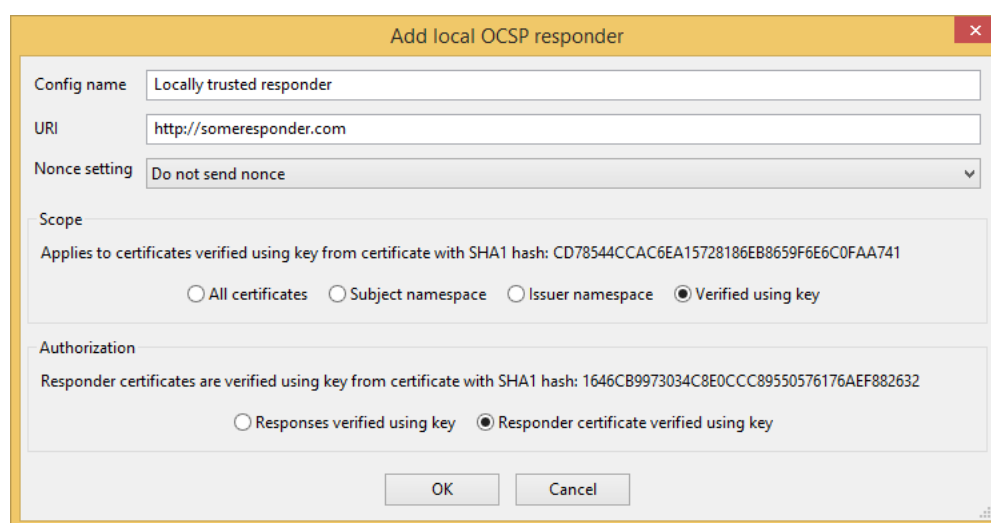


Figure 32 Local OCSP responder configuration

To add a certificate to the blacklist, select **Blacklist** in the **Type** field then click the **Add certificate to blacklist** option. To remove a certificate from the list, select the certificate and click the **Remove certificate from blacklist**.

To add a certificate to the no check list, select **No check** in the **Type** field then click the **Add certificate** button. A dialog similar to the dialog shown below will be displayed. Browse to the certificate then choose whether to add only the specified certificate or also certificates subordinate to this certificate. Note the whitelist option is certificate based, not public key based. Thus if multiple certificates are issued for the same key and all should be whitelisted, each must be added.

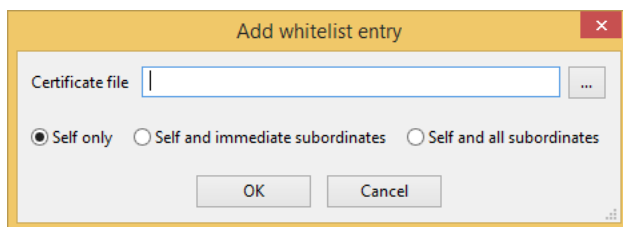


Figure 33 Whitelist entry configuration

### **Certificates**

The **Certificates** tab displays a list of certificate files present in the **Certs folder** specified on the **Path Building Settings** tab. The list displays the subject name, issuer name, the SHA1 hash of the DER encoded certificate and filename for each certificate. The list can be sorted on each column in either ascending or descending order by clicking the column label. To import a certificate or folder of certificates, click the **Import Certificate** button or **Import Certificates** button. To delete a certificate file, select the certificate to delete then click the **Delete Selected Certificates** button. To delete non-CA certificates, click the **Delete non-CA certificates** button. The list is populated when the **Edit path settings** notebook is displayed. After changing the **Certs folder** setting, dismiss and re-launch the **Edit path settings** notebook to refresh the **Certificates** list.

Care should be exercised when deleting certificate files that were downloaded automatically (i.e., the files named using a hash). The **URIs** tab displays the last modified time retrieved from the server when a certificate (or PKCS #7 or CRL) file was downloaded. If the local copy of the file is deleted and the URI is not deleted from the URI database, a fresh copy of the certificate may not be re-downloaded.



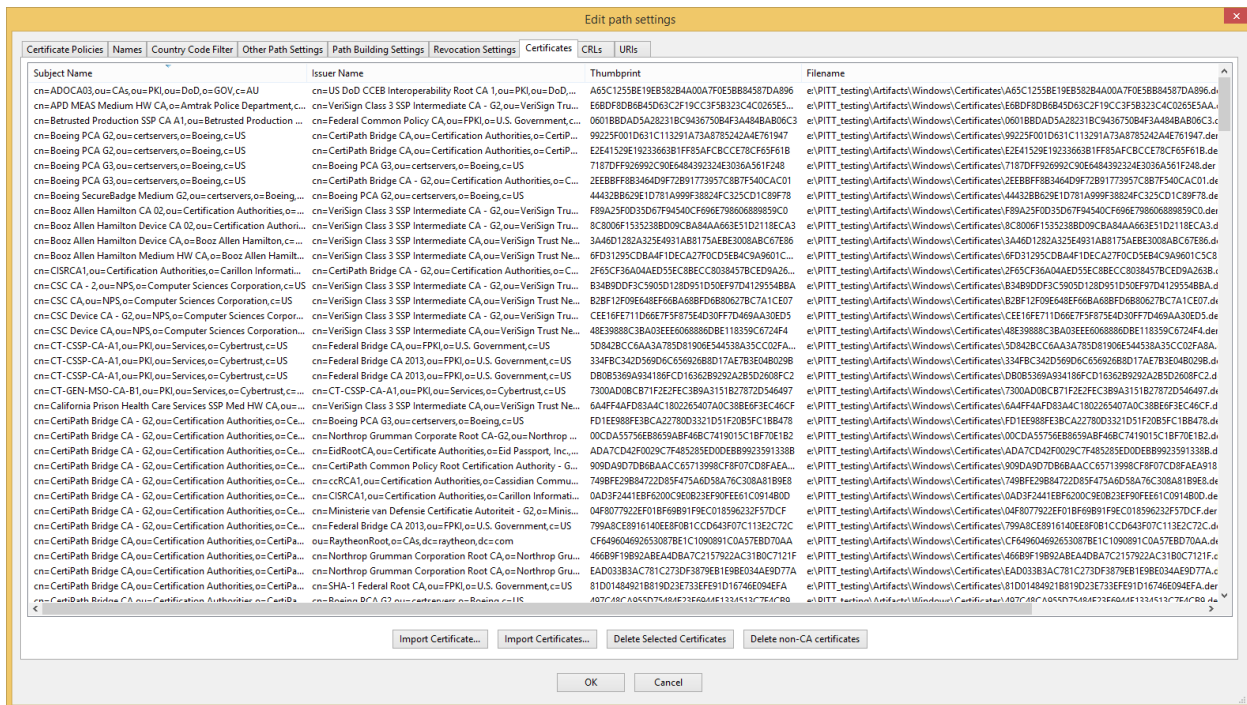


Figure 34 Certificates list

## CRLs

The **CRLs** tab displays a list of CRL files present in the **CRLs folder** specified on the **Revocation Settings** tab. The list displays the issuer name, distribution point, next update and filename for each CRL. The list can be sorted on each column in either ascending or descending order by clicking the column label. To import a CRL or folder of CRLs, click the **Import CRL** button or **Import CRL** button. To delete a CRL file, select the CRL to delete then click the **Delete Selected CRLs** button. To delete stale CRLs, click the **Delete stale CRLs** button. The list is populated when the **Edit path settings** notebook is displayed. After changing the **CRLs folder** setting, dismiss and re-launch the **Edit path settings** notebook to refresh the **CRLs** list.

Care should be exercised when deleting CRL files that were downloaded automatically (i.e., the files named using a hash). The **URIs** tab displays the last modified time retrieved from the server when a CRL (or PKCS #7 or certificate) file was downloaded. If the local copy of the file is deleted and the URI is not deleted from the URI database, a fresh copy of the certificate may not be re-downloaded.

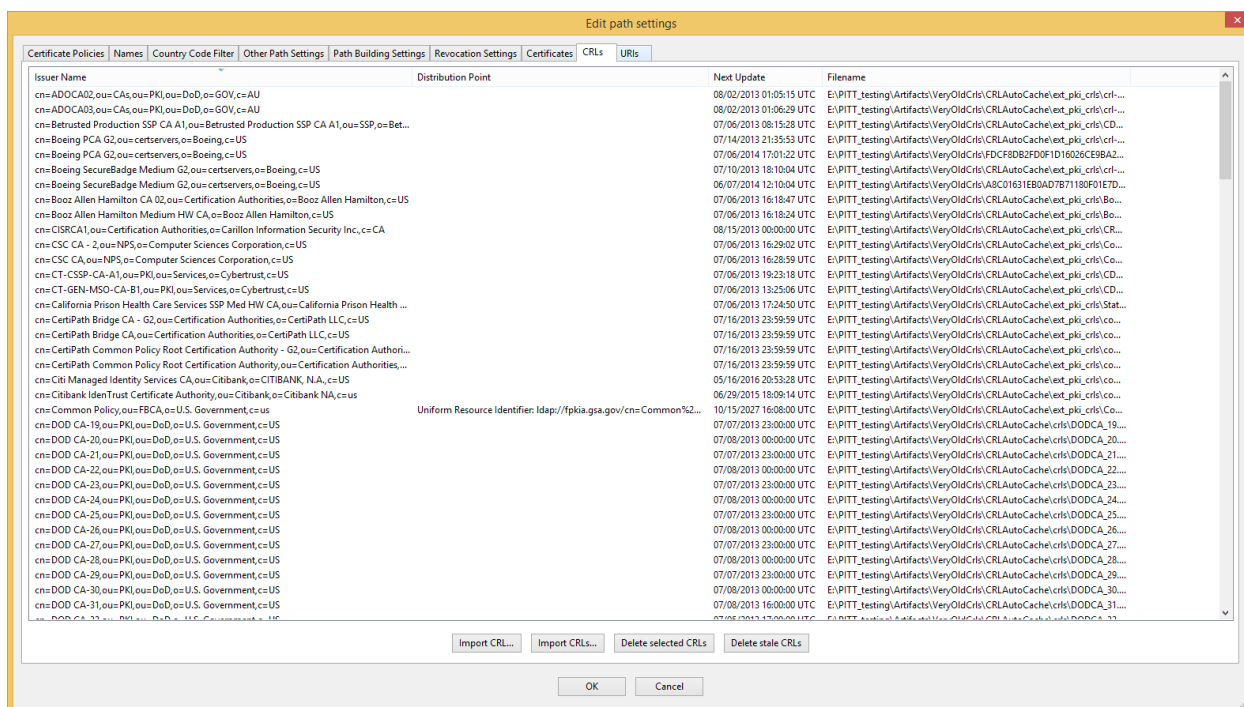


Figure 35 CRLs

## URIs

The **URIs** tab displays a list of URIs read from the uriLastModified.db file present in the **Certs folder** and **CRLs folder** locations. The list includes the URI and last modified time returned from the server. The last modified value is used to avoid downloading artifacts that have not changed since a previous download operation. As noted above, when manually deleting automatically downloaded certificates or CRL files, the uriLastModified.db file should either be deleted or the URI associated with the manually deleted file should be deleted from the uriLastModified.db list.

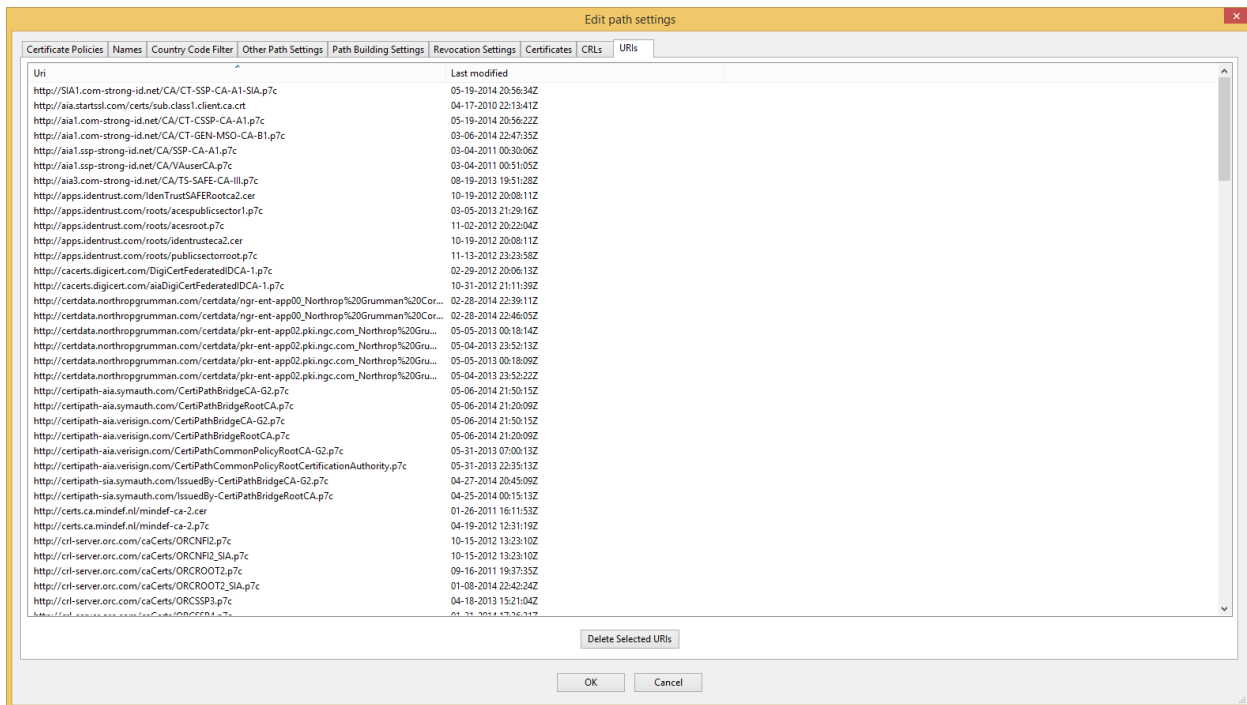


Figure 36 Last modified times for URIs

## Edit security environments

Security environments contain settings that are less volatile. These settings include:

- TACT trust anchor store to use during certification path validation
- Certificate policies database to use during path settings configuration and logging of certification path validation results
- Cryptographic algorithm prohibitions

To specify a TACT trust anchor store or certificate policies database, use the **PKI Artifacts** tab in the security environment editor dialog, as shown below. To specify a TA store or policies database, click the ... button adjacent to the corresponding field and navigate to the desired file. On Windows, TA stores are typically stored in `C:\TactSettings\Tas` and policies databases are stored in `C:\TactSettings`. On Linux, the usual locations are `/etc/tact/tas` and `/etc/tact`, respectively.

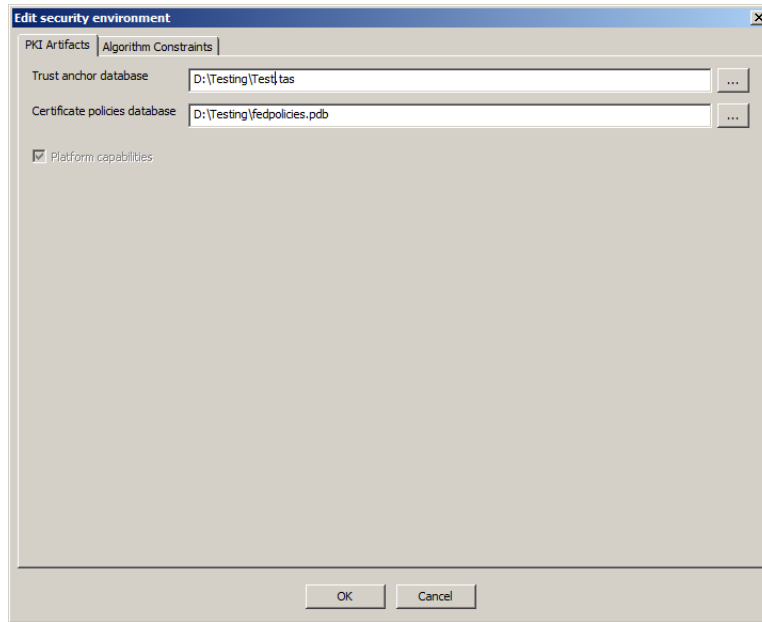


Figure 37 TA store and certificate policies database specification

To specific cryptographic algorithm prohibitions or to define minimum acceptable cryptographic key sizes, use the **Algorithm Constraints** tab as shown below. To prohibit an algorithm, select the algorithm in the left list box and click the -> button (or double click the algorithm). To remove an algorithm prohibition, select the algorithm in the right list box and click the <- button (or double click the algorithm). To specify a minimum key size, choose a value from the drop list corresponding to the algorithm of interest.

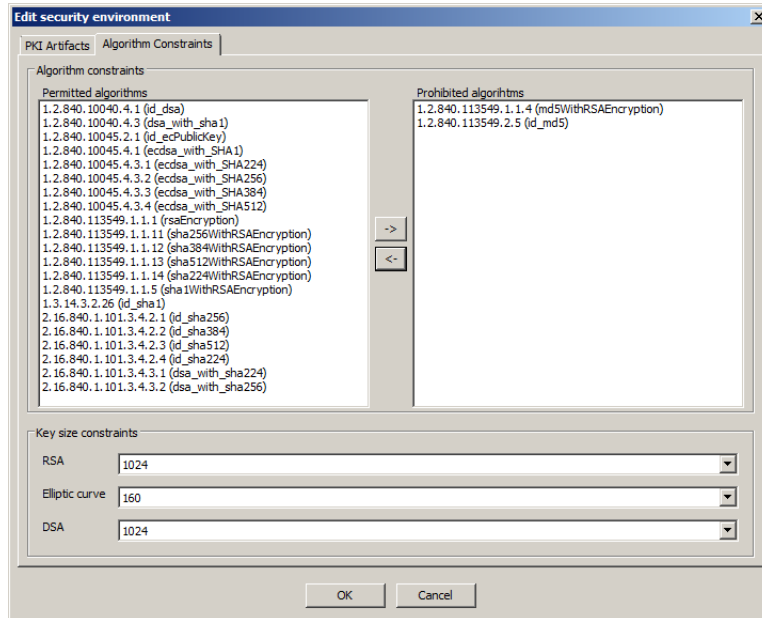


Figure 38 Defining algorithm prohibitions and minimum key sizes

Beginning with TACT v1.2, security environment definitions included a server blacklist and a connection timeout, as shown below.

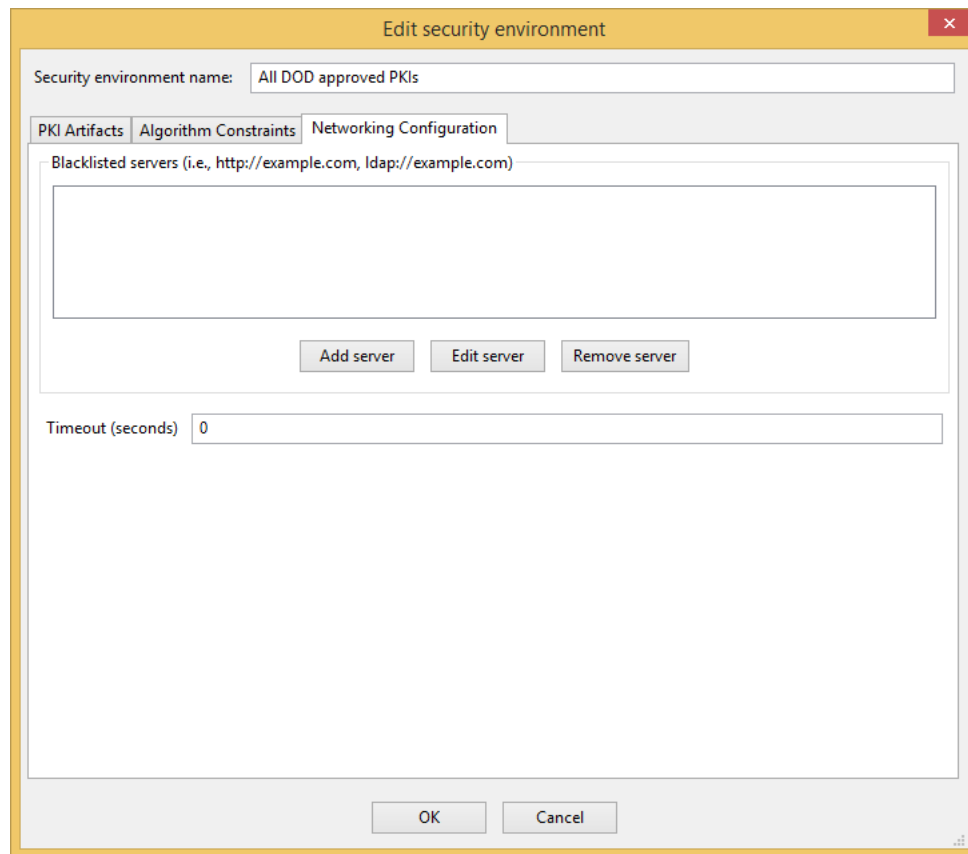


Figure 39 Networking configuration panel

These settings are infrequently used. The blacklisted servers list can be used to avoid attempting to connect to servers that have been decommissioned or that provide unacceptable performance. The host name is used for blacklist purposes. Scheme values are not considered. By default (i.e., without specifying a value in the security environment), the connection timeout value is 30 seconds.

## Edit certificate policies databases

The certificate policies database editor, shown below, can be accessed via the TACT Server Configuration utility options dialog or the TA Store Manager utility options dialog.

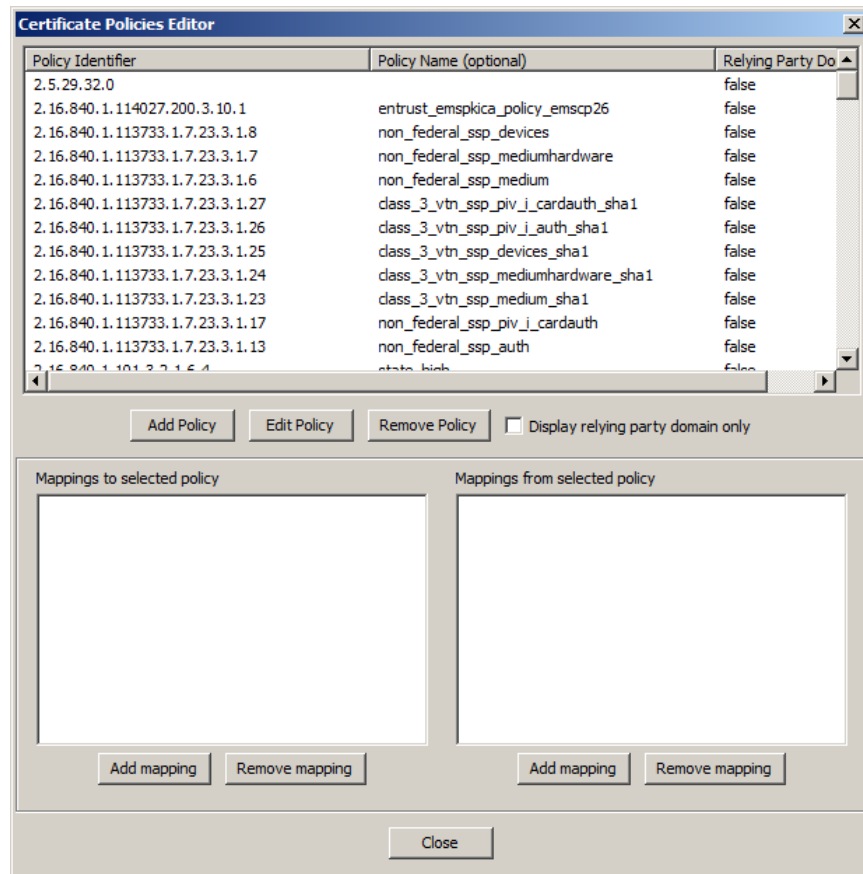
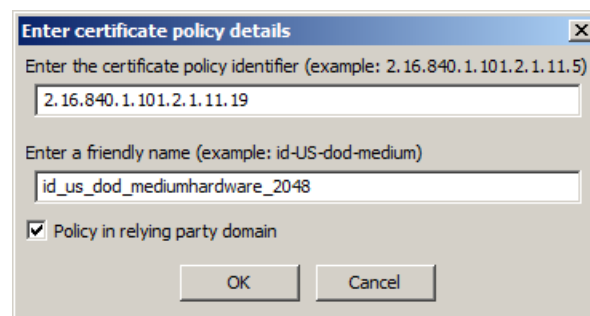


Figure 40 Certificate policies database editor

The certificate policies database editor provides a means for editing the contents of a certificate policies database. The schema of the database is provided in [Appendix B](#). To add a new policy, click the **Add Policy** button. To edit a policy that is already present in the database, select the policy in the list then click the **Edit Policy** button. The figure below shows the dialog used when a new certificate policy is added or a certificate policy is edited.



### Figure 41 Certificate policy definition

## Edit logging configuration

Logging configurations can be edited using the Log Configuration Editor dialog, shown below.

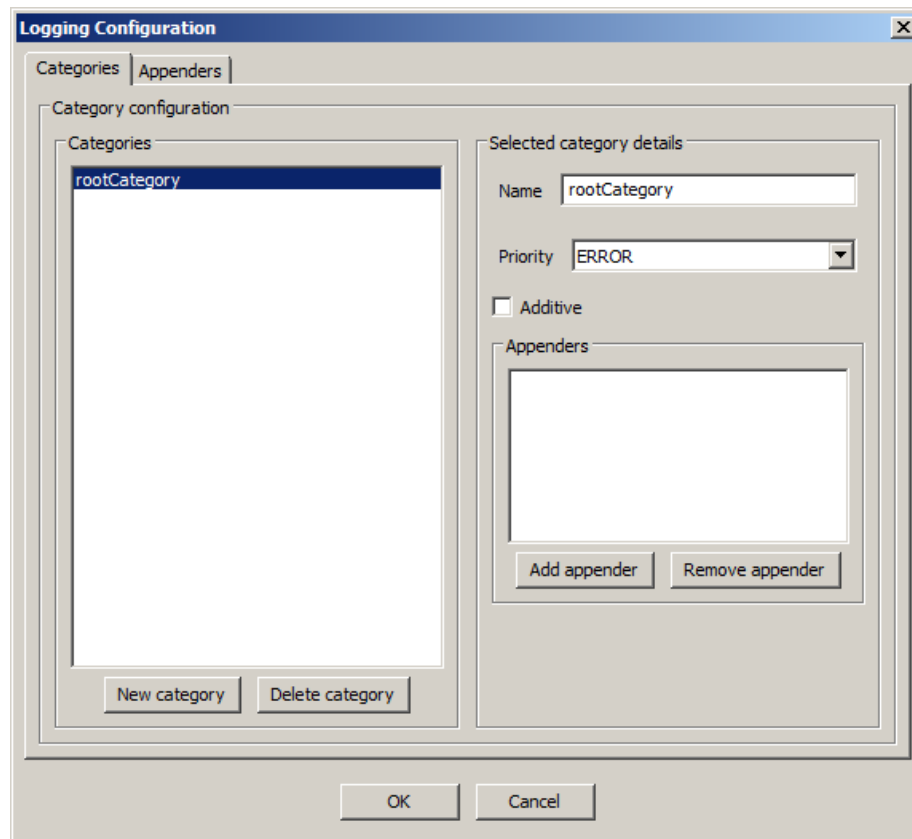


Figure 42 Logging configuration editor

This dialog enables the definition multiple logging categories and appenders. Categories are used by applications to indicate a logging source. Appenders are destinations for log information. Zero or more appenders can be associated with a category. At least one category, `rootCategory`, must be defined. Additional categories may be defined to direct output from different log sources to different destinations. TACT management applications use the following logging categories:

- RhUtils
- TaStoreManager
- TactServerConfig
- TactComplianceAssessment
- TactCli

TACT plugins use the following logging categories:

- RhUtils
- tact\_auth\_core
- tact\_auth\_core\_access
- TactPlugin
- CertLogDatabase

The basic steps of preparing a logging configuration are as follows:

- Identify the log categories of interest
- Define appenders for log destinations
- Define categories of interest and associated the corresponding appender

The following steps illustrate how to use the logging configuration editor to prepare a logging configuration file for use by the various management utilities, with each application's output begin written to an independent file, RhUtils output being written to a rolling set of files and TactCli output being written both to a file and the console.

To define an appender to direct log output to a file, perform the following steps:

- Click on the **Appenders** tab
- Click the **New appender** button
- Enter a name for the new appender in the **Name** field
- Select **File appender** from the **Type** drop list
- Enter the full path and filename of the file to receive log information
- Choose a **Layout** (the default Pattern layout is recommended)

The figure below shows the appender details after following the steps above.



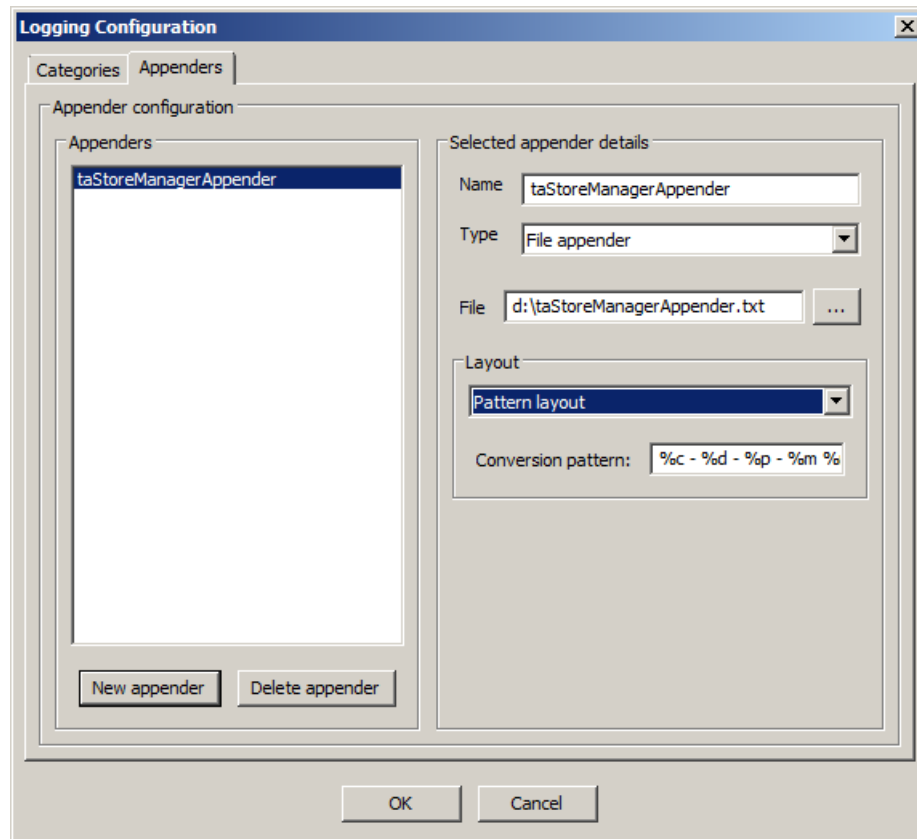


Figure 43 New file appender definition

To associate the new appender with a category to receive TA Store Manager output, perform the following steps:

- Click on the **Categories** tab
- Click the **New category** button
- Enter TaStoreManager in the **Name** field
- Select the desired **Priority** value
- Click the **Add Appender** button
- Select the newly defined appender from the drop list and click **OK**

The figure below shows the category details after following the steps above. In this example, the **Additive** check box is not checked. When this box is checked, the log output is written to each appender associated with the category and to any appenders associated with the rootCategory. In this example, the output is not written to the rootCategory.

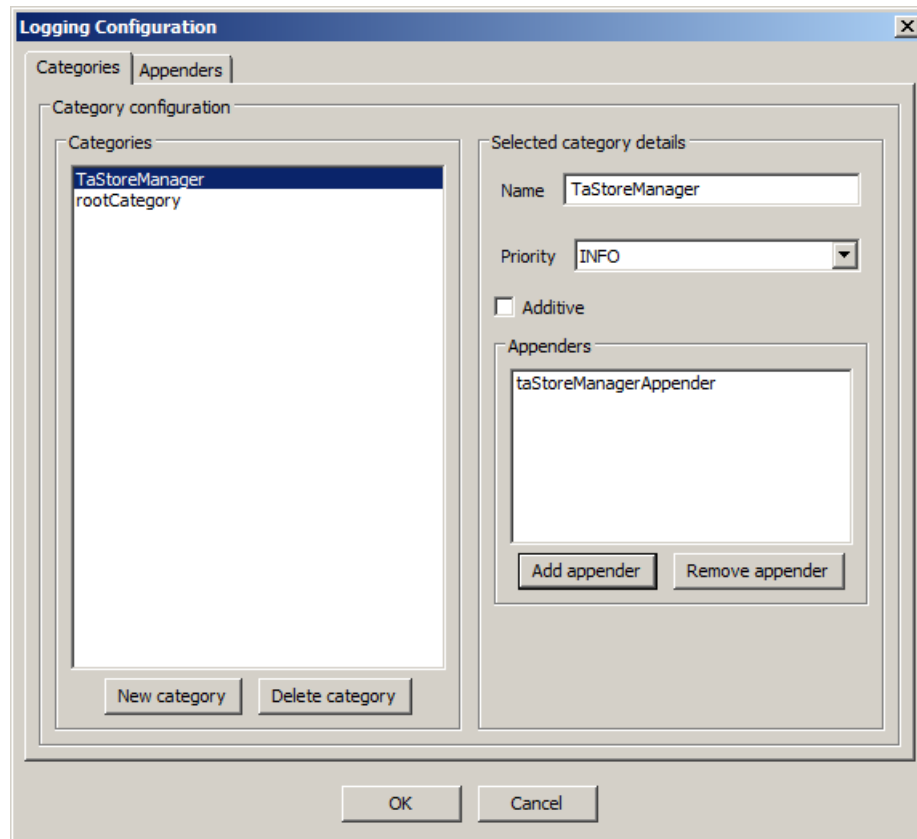


Figure 44 New category definition

Repeat the steps defined above for creating new file appenders for the TACT Server Configuration utility, TACT Compliance Assessment utility and TACTCli utility. Since TactCli output is to also be written to the console, perform the following steps to prepare a new console appender.

- Click on the **Appenders** tab
- Click the **New appender** button
- Enter a name for the new appender in the **Name** field
- Select **Console appender** from the **Type** drop list
- Choose a **Layout** (Simple layout is recommended for console appenders)

The figure below shows the appender details following execution of the above steps.

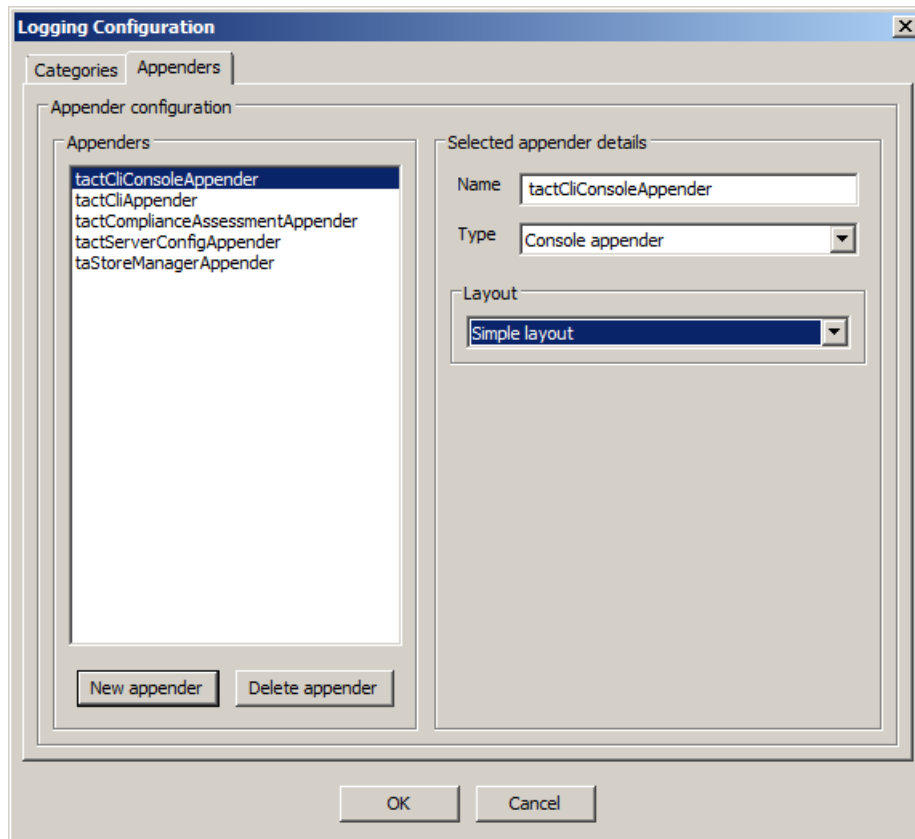


Figure 45 New console appender definition

In the above figure, there is one file appender for each application plus a console appender for TactCli. To associate the new console appender with the TactCli category, follow the steps described above for associating appenders with categories. In this case, the category will feature two appenders, as shown below.

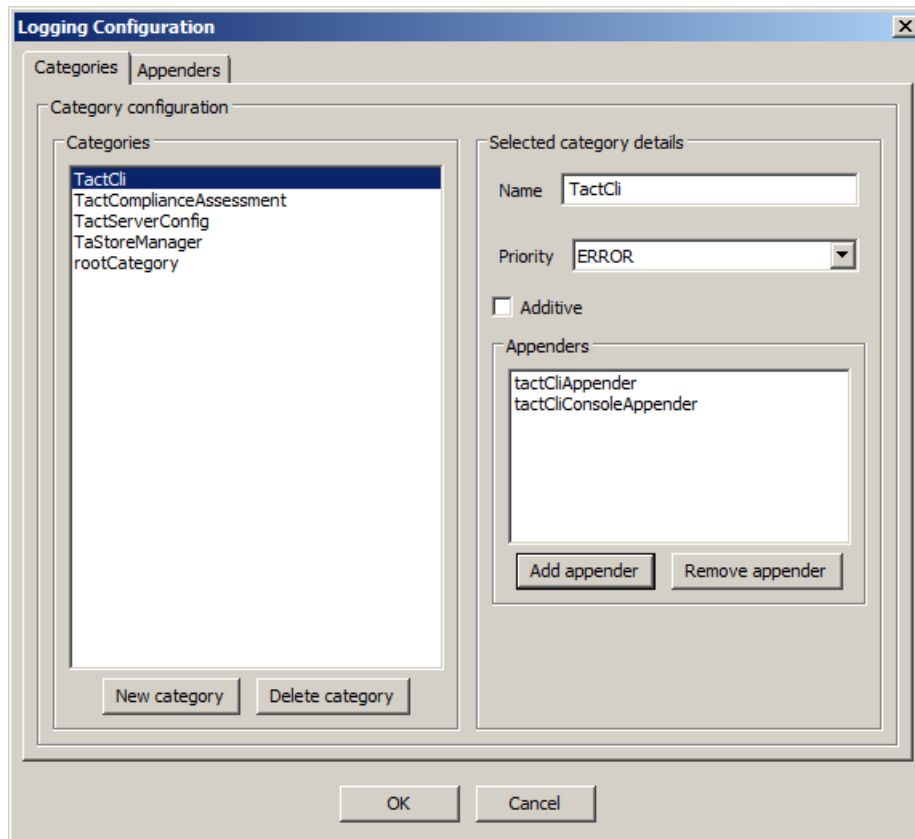


Figure 46 Category definition with multiple appenders

The last steps are to configure RhUtils output to use a rolling file appender and to, optionally, define an appender for the rootCategory. The figure below shows an example appender definition for a rolling file appender with log file size limited to 5MB.

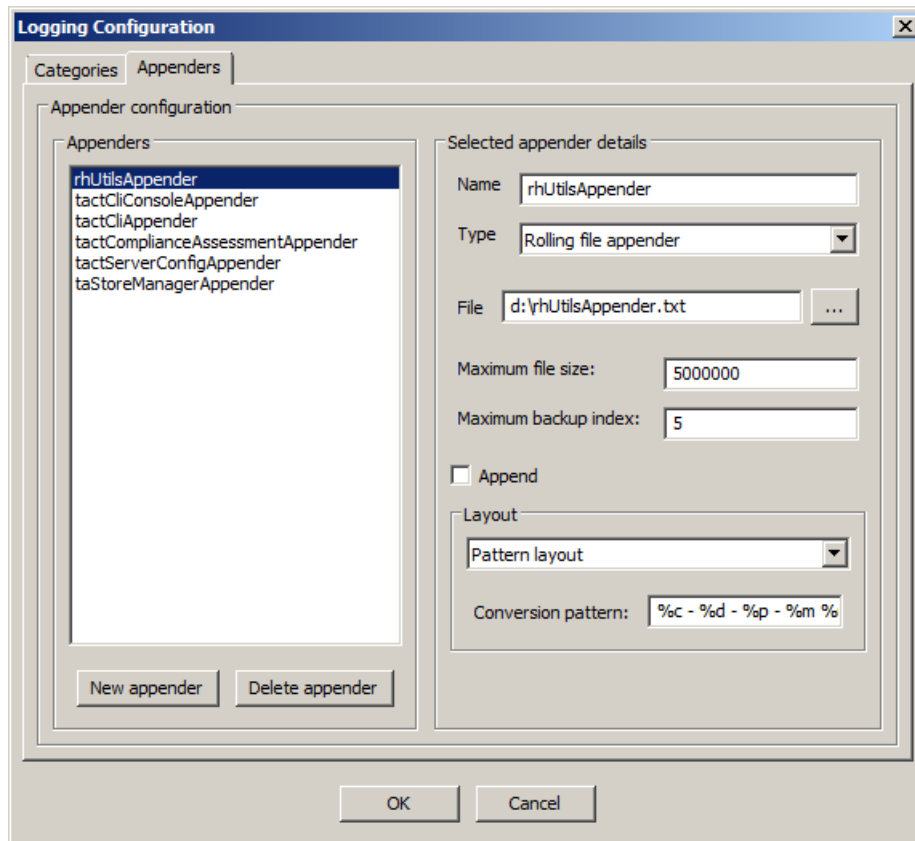


Figure 47 New rolling file appender definition

Using the steps above, each management application can share a common logging configuration file with output directed to independent files. [Appendix D](#) contains the textual configuration corresponding to the above steps (augmented with a rolling file appender for rootCategory).

An alternative, simpler approach would be to create a logging configuration file per application and define an appender for the rootCategory only, to capture all logging output from the application in one location. The logging configuration approach selected will vary with TACT operator preferences.

In all cases, it is recommended that only the ERROR priority be used for operational situations except for limited troubleshooting periods as DEBUG output can be fairly noisy.

Note, logging configurations may contain platform specific mechanisms, i.e., Event log appender and Win32 debug appender on Windows or Syslog appender on Linux. Logging configurations containing platform specific mechanisms must only be edited or used on the corresponding platform.

## Server Management

TACT plugins are designed to require very little additional management effort beyond that which is required to maintain a PKI-enabled web server. The primary considerations involved in managing a server running TACT are system security settings and log management.

### System Security Settings

TACT plugin security depends on managing access to both Trust Anchor stores and to TACT settings databases. These files contain no secret data; it is safe to allow even unprivileged users to read them. Only users with administrative control of the server should be granted write access to these files. On systems where the TA store is managed by a different entity than the server configuration, write permissions should reflect this separation.

TACT log files and log databases, on the other hand, may be considered sensitive with regard to read access. It is a good practice to limit this access to system administrators and the account under which the web service (which must be granted read/write access to these files in order to update them).

On Windows servers, the account running the application pool (server 2008) will require read-only access to the following directories:

- The directory containing the TACT settings database (typically `C:\TactSettings`)
- The directory containing the TACT policy database (typically `C:\TactSettings`)
- The directory containing the TA stores (typically `C:\TactSettings\Tas`)

The web server account will also require read **and** write access to the directory containing the log database and the directory containing the log files, if the plugin is configured to log information to files. Due to the potential size of these files, it is common to store them on a different drive or partition to prevent administrator error from leading to a full system drive.

On Linux servers, the account running httpd will require read-only access to the following directories:

- The directory containing the TACT settings database (typically `/etc/tact`)
- The directory containing the TACT policy database (typically `/etc/tact`)
- The directory containing the TA stores (typically `/etc/tact/tas`)

The same account will require read **and** write access to the directory containing the log database and the directory containing log files, if the plugin is configured to write log data to files. These log files and the PKI log database will typically be written to `/var/log/tact`.

For systems where SELinux is enabled, filesystem ACLs are insufficient. The labels on these files and directories must also be configured appropriately, or `httpd` will be denied access even if the file permissions appear correct. Typically the configuration data for `httpd` (including both the module config in `/etc/httpd` and the TACT configuration in `/etc/tact`) will need to be labeled `httpd_config_t` and the log data will need to be labeled `httpd_log_t`. `mod_auth_certtrust.so` will need to be labeled `httpd_modules_t`. The installer prints commands that the server operator can use to apply these settings.

### Using TACT Management Tools

Because the TACT configuration files are expected to be secured on production systems, in order to edit them on a live system, the management tools will need to be run with elevated privileges. On Linux systems, this means either logging into the desktop as root and choosing the tools from the applications menu, or using another utility to elevate privileges prior to launching the tools. In a default install, all four of the tools have symbolic links in `/usr/local/bin`. This should be on the path for most users, for ease of use with `sudo` or `su -c`. On Windows systems, this means launching the applications as administrator.

### File size considerations

TACT plugins do not impose any arbitrary limits on log file sizes. For textual log files, it is generally a good practice either to use a rolling file appender or to configure the plugin to send log messages to some externally managed log, such as the event log or `syslog`. Most installations will prefer the rolling file configuration, as this caps the size of individual files, preserves older logs, up to a configurable limit, for offline analysis, and prevents interference with the system logging facility.

The PKI log database can grow without bound. Once it begins to approach the size of the available memory on the machine, there may be some impact on server performance. The growth of this database is generally slow, however, as certificates are only added if they are not already present. The number of active credentials visiting the site will be the primary determinant of log database size, and server administrators should rotate this during maintenance windows if it becomes a concern.

## Environmental considerations

TACT plugins need access to the full certification path validated by the web server in order to apply appropriate constraints. The three supported servers vary slightly in their ability to supply this information to TACT.

### IIS

TACT can use Microsoft's Cryptographic Application Programming Interface (CAPI) on Windows operating systems to retrieve a certification path trusted by the web server, but unless the CA certificates are stored in the machine's CAPI intermediate certificates store, TACT will need to use CAPI in a way that may cause network access. TACT will store the certificates retrieved in this manner in a CAPI on-disk store next to its PKI log database. For best performance, however, it is recommended that server operators install those intermediate CA certificates.

### Apache/mod\_nss

When mod\_nss is used for TLS, it will only supply intermediate CA certificates to TACT if those are stored in mod\_nss' certificate database. Server operators will need to keep the mod\_nss security database updated with CA certificates as described in the mod\_nss section of the Apache PKE guide.

### Apache/mod\_ssl

When mod\_ssl is used for TLS, TACT has full access to the certification path so long as it has been configured with a pointer to the mod\_ssl CACertificate file. Administrators will need to ensure that, if the CACertificate file is moved after installation, TACT is also updated with the new file location.

For both mod\_ssl and mod\_nss, TACT maintains a cache of recently validated certificates. The cache is stored in the same location as the PKI log database, and is emptied whenever the server is started. The cache is used only to work around certain session management behaviors in the TLS plugins in combination with some clients. If this cache ever grows large, it can be reset simply by restarting the httpd service with no adverse impact on the user experience.



## Appendix A: Support

### Web Site

Please visit the URL below for additional information.

<http://iase.disa.mil/pki-pke>

### Technical Support

Contact technical support at the email address below.

[dodpke@mail.mil](mailto:dodpke@mail.mil)

## Appendix B: File Formats

TACT uses a number of different types of files for configuration. The table below provides a summary overview of these different file types. The following sections provide additional detail for selected file types.

File type	File extensions	Description
TACT Settings	*.sdb	Contains path settings, security environment and protected resource definitions. Primarily edited using the TACT Server Configuration utility.
TACT Trust Anchor Store	*.tas	Contains trust anchors. Edited using the TA Store Manager utility and TactCli utility (when using TAMP messages).
Certificate Policies	*.pdb	Contains certificates policies, certificate policy descriptions and policy mappings. Primarily edited using the TA Store Manager utility or TACT Server Configuration utility.
Path Settings	*.ps	Contains path settings definition
Security Environment	*.se	Contains security environment definition
TACT Resource	*.tr	Contains protected resource definition
Logging Configuration	*.conf; *.properties	Contains logging configuration
PKI Log	*.db	Contains certificates, certification paths and other session-related information
URI last modified	uriLastModified.db	Automatically generated file that maintains the last modified time for resources obtained from an indicated URI
CRL file info	crlIndex.db	Automatically generated file that maps CRL information to a file name
Trust Anchors	*.ta	Contains an RFC 5914 TrustAnchorChoice object
X.509 Certificates	*.der; *.crt; *.cer	Contains an RFC 5280 Certificate

		object
TAMP Update	*.tur; *.tampUpdate	Contains an RFC 5934 TAMPUpdate message (wrapped in a ContentInfo or SignedData)
TAMP Status Response	*.tsr; *.statusResponse	Contains an RFC 5934 TAMPStatusResponse message (wrapped in a ContentInfo or SignedData)

TACT utilities place limit the size of each file type. Prior to opening a file, the size is checked and if the limit is exceeded an error message is displayed or logged. The following limits apply, each of which is significantly larger than expected operational sizes:

- TACT Settings: 25MB
- TACT Trust Anchor Store: 10MB
- Certificate Policies: 10MB
- Path Settings: 2MB
- Security Environment: 2MB
- Logging Configuration: 2MB
- Trust Anchors: 2MB
- X.509 Certificates: 2MB
- TAMP Update: 10MB
- TAMP Status Response: 10MB

## TACT Settings

TACT settings databases contain all configuration information for a TACT plugin installation, including certification path validation settings, basic security environment settings and protected resource definitions. TACT settings databases are SQLite databases that conform to the schema shown below. This schema is provided for informational purposes only. TACT settings databases should only be edited using the provided utilities.

```
CREATE TABLE PathSettings (
    PathSettingsId INTEGER PRIMARY KEY AUTOINCREMENT,
    PathSettingsName TEXT UNIQUE,
    PathSettings BLOB,
    LastChange DATETIME);
CREATE TABLE SecurityEnvironments (
    SecurityEnvironmentId INTEGER PRIMARY KEY AUTOINCREMENT,
    SecurityEnvironmentName TEXT UNIQUE,
    SecurityEnvironment BLOB,
    LastChange DATETIME);
CREATE TABLE Resources (
```

```

    ResourceId INTEGER PRIMARY KEY AUTOINCREMENT,
    ResourceName TEXT UNIQUE,
    PathSettings INTEGER OPTIONAL,
    SecurityEnvironment INTEGER OPTIONAL,
    LastChange DATETIME,
    FOREIGN KEY(PathSettings) REFERENCES PathSettings(PathSettingsId) ON DELETE SET NULL,
    FOREIGN KEY(SecurityEnvironment) REFERENCES
SecurityEnvironments(SecurityEnvironmentId) ON DELETE SET NULL);
PRAGMA user_version=1;
PRAGMA foreign_keys = ON;
CREATE TRIGGER [delete_PathSettings]
    BEFORE DELETE ON [PathSettings]
    FOR EACH ROW
    BEGIN
        UPDATE Resources SET PathSettings = NULL WHERE Resources.PathSettings =
old.PathSettingsId;
    END;
CREATE TRIGGER [delete_SecurityEnvironments]
    BEFORE DELETE ON [SecurityEnvironments]
    FOR EACH ROW
    BEGIN
        UPDATE Resources SET SecurityEnvironment = NULL WHERE Resources.SecurityEnvironment =
old.SecurityEnvironmentId;
    END;

```

TACT v1.2 introduced several new tables to the TACT settings database. The new tables are listed below. Note, comparison operations performed using TACT Compliance Assessment or TactCli do not compare the contents of the new database tables. Similarly, comparison operations does not address the contents of folders identified by TACT settings.

```

CREATE TABLE RevocationFreshnessConfigs (
    FreshnessId INTEGER PRIMARY KEY AUTOINCREMENT,
    FreshnessName TEXT UNIQUE,
    MechType INTEGER,
    Priority INTEGER,
    GracePeriod INTEGER,
    Freshness INTEGER,
    CertificateToValidateArtifact BLOB OPTIONAL,
    ScopeType INTEGER,
    PathSettingsId INTEGER,
    FOREIGN KEY(PathSettingsId) REFERENCES PathSettings(PathSettingsId) ON DELETE SET
NULL,
    UNIQUE(FreshnessName, PathSettingsId));
CREATE TABLE Certificates (
    CertificateId INTEGER PRIMARY KEY AUTOINCREMENT,
    CertificateHash TEXT NOT NULL UNIQUE,
    Certificate BLOB);
CREATE TABLE Blacklist (
    BlacklistEntryId INTEGER PRIMARY KEY AUTOINCREMENT,
    CertificateId INTEGER,
    PathSettingsId INTEGER,
    FOREIGN KEY(PathSettingsId) REFERENCES PathSettings(PathSettingsId) ON DELETE SET

```

```

NULL,
    FOREIGN KEY(CertificateId) REFERENCES Certificates(CertificateId) ON DELETE SET
NULL,
    UNIQUE(CertificateId, PathSettingsId));
CREATE TABLE Ignorelist (
    IgnorelistEntryId INTEGER PRIMARY KEY AUTOINCREMENT,
    CertificateId INTEGER,
    PathSettingsId INTEGER,
    FOREIGN KEY(PathSettingsId) REFERENCES PathSettings(PathSettingsId) ON DELETE SET
NULL,
    FOREIGN KEY(CertificateId) REFERENCES Certificates(CertificateId) ON DELETE SET
NULL,
    UNIQUE(CertificateId, PathSettingsId));
CREATE TABLE Whitelist (
    WhitelistEntryId INTEGER PRIMARY KEY AUTOINCREMENT,
    ScopeOfWhitelist INTEGER,
    CertificateId INTEGER,
    PathSettingsId INTEGER,
    FOREIGN KEY(PathSettingsId) REFERENCES PathSettings(PathSettingsId) ON DELETE SET
NULL,
    FOREIGN KEY(CertificateId) REFERENCES Certificates(CertificateId) ON DELETE SET
NULL,
    UNIQUE(CertificateId, PathSettingsId));
CREATE TABLE LocalOcsp (
    LocalOcspId INTEGER PRIMARY KEY AUTOINCREMENT,
    LocalOcspName TEXT UNIQUE,
    Priority INTEGER,
    Uri TEXT,
    NonceSetting INTEGER,
    ScopeCertOrName BLOB OPTIONAL,
    ScopeType INTEGER,
    AuthCert BLOB,
    AuthType INTEGER,
    PathSettingsId INTEGER,
    FOREIGN KEY(PathSettingsId) REFERENCES PathSettings(PathSettingsId) ON DELETE SET
NULL,
    UNIQUE(LocalOcspName, PathSettingsId));
PRAGMA user_version=2;

```

## TACT Trust Anchor Store

TACT trust anchor stores contain trust anchors for use by TACT plugins. TACT trust anchor stores are SQLite databases that conform to the schema shown below. TACT v1.0 uses only the TrustAnchors and SequenceNumbers tables. This schema is provided for informational purposes only. TACT trust anchor stores should only be edited using the Trust Anchor Store Manager utility.

```

CREATE TABLE TrustAnchors (
    TrustAnchorId INTEGER PRIMARY KEY AUTOINCREMENT,
    SubjectPublicKeyInfo BLOB,
    TrustAnchor BLOB,

```

```

        Name TEXT,
        SKID TEXT UNIQUE);
CREATE TABLE SequenceNumbers (
    SequenceNumberId INTEGER PRIMARY KEY AUTOINCREMENT,
    SKID TEXT UNIQUE,
    SequenceNumber INTEGER);
CREATE TABLE TaStoreAdminCertificates (
    CertificateId INTEGER PRIMARY KEY AUTOINCREMENT,
    Certificate BLOB);
CREATE TABLE StoreInformation (
    TrustAnchorStoreName TEXT,
    ApexTrustAnchor BLOB OPTIONAL,
    MaxNumberOfTrustAnchors INTEGER OPTIONAL);
PRAGMA user_version=1;

```

## Certificate Policies

The certificate policies database can be used to simplify configuration of certificate policy related constraints by providing textual descriptions in addition to object identifiers. Additionally, configuration can be performed in terms of certificate policies from the enterprise of the administrator. The policies database is a SQLite database that conforms to the schema shown below. In some environments, creation of a certificate policies database may be performed using custom SQLite scripts or tools to import existing policy information into a new database instance.

```

CREATE TABLE CertificatePolicies (
    CertPolicyId INTEGER PRIMARY KEY AUTOINCREMENT,
    PolicyOid TEXT UNIQUE,
    InRelyingPartyDomain BOOL,
    PolicyName TEXT OPTIONAL);
CREATE TABLE PolicyMappings (
    IssuerDomain INTEGER,
    SubjectDomain INTEGER,
    PRIMARY KEY (IssuerDomain, SubjectDomain),
    FOREIGN KEY(IssuerDomain) REFERENCES CertificatePolicies(CertPolicyId) ON DELETE
    CASCADE,
    FOREIGN KEY(SubjectDomain) REFERENCES CertificatePolicies(CertPolicyId) ON DELETE
    CASCADE);
PRAGMA user_version=1;
PRAGMA foreign_keys = ON;
CREATE TRIGGER [delete_CertificatePolicy]
    BEFORE DELETE ON [CertificatePolicies]
    FOR EACH ROW
    BEGIN
        DELETE FROM PolicyMappings WHERE PolicyMappings.IssuerDomain = old.CertPolicyId;
        DELETE FROM PolicyMappings WHERE PolicyMappings.SubjectDomain = old.CertPolicyId;
    END;

```

## Path Settings

Path settings configuration objects contain various settings that influence certification path validation. Path settings configuration files contain XML generated and processed by Boost serialization components. These files should not be edited manually.

## Security Environment

Security environment configuration objects contain various settings that influence certification path validation. Security environment values are generally more broadly applicable than those contained in path settings configuration objects. Security environment configuration files contain XML generated and processed by Boost serialization components. These files should not be edited manually.

## TACT Resource

TACT resource configuration objects contain various settings that influence access control decisions. TACT resource configuration files contain XML generated and processed by Boost serialization components. These files should not be edited manually.

## Logging Configuration

TACT uses the log4cpp library for logging. The log4cpp library uses a configuration file similar to the file used by the log4j library. While these files are typically hand edited, TACT provides a graphical user interface for editing these. It is strongly recommended that these files be edited using the provided tools. In some cases, especially when moving logging configuration files from one platform to another, it may be necessary to edit the files using a text editor to remove platform specific mechanisms.

## PKI Log

The PKI log is automatically populated by the TACT plugin during operation. The PKI log is a SQLite database that conforms to the schema shown below. The schema is provided to enable the creation of scripts and other analysis tools for use in reviewing PKI log contents. No analysis tool is provided with TACT.

```
CREATE TABLE Certificates (  
    CertificateId INTEGER PRIMARY KEY AUTOINCREMENT,  
    Certificate BLOB,  
    IssuerName TEXT,  
    NotBefore TEXT,  
    NotAfter TEXT,  
    SubjectName TEXT,  
    CertHash TEXT NOT NULL UNIQUE,  
    isCA INTEGER,  
    SKID TEXT,  
    AKID TEXT OPTIONAL,
```

```

        certificatePolicies TEXT OPTIONAL,
        certificateType TEXT OPTIONAL,
        edipi TEXT OPTIONAL
    );
CREATE TABLE AssociatedPaths (
    PartialCertificationPath NUMERIC,
    Certificate NUMERIC,
    PRIMARY KEY (PartialCertificationPath, Certificate),
    FOREIGN KEY(PartialCertificationPath) REFERENCES
PartialCertificationPaths(PartialCertificationPathId),
    FOREIGN KEY(Certificate) REFERENCES Certificates(CertificateId)
);
CREATE TABLE PartialCertificationPaths (
    PartialCertificationPathId INTEGER PRIMARY KEY AUTOINCREMENT,
    PathHash TEXT,
    NumCerts NUMERIC
);
CREATE TABLE PartialCertificationPathMembers (
    PartialCertificationPath NUMERIC,
    Certificate NUMERIC,
    PathIndex NUMERIC,
    PRIMARY KEY (PartialCertificationPath, Certificate, PathIndex),
    FOREIGN KEY(PartialCertificationPath) REFERENCES
PartialCertificationPaths(PartialCertificationPathId),
    FOREIGN KEY(Certificate) REFERENCES Certificates(CertificateId)
);
CREATE TABLE AccessLog (
    EventId INTEGER PRIMARY KEY AUTOINCREMENT,
    Time TEXT,
    SourceIP TEXT,
    SourcePort INTEGER,
    DestIP TEXT,
    DestPort INTEGER,
    UrlAccessed TEXT,
    AccessAllowed INTEGER,
    CertId NUMERIC,
    PathId NUMERIC OPTIONAL,
    FOREIGN KEY(CertId) REFERENCES Certificates(CertificateId),
    FOREIGN KEY(PathId) REFERENCES PartialCertificationPaths(PartialCertificationPathId)
);
PRAGMA user_version=1;

```

## URI Last Modified

The UriLastModified database is populated when a resource is successfully downloaded from a server via HTTP. If the server provides a last modified time for the requested resource, the value is stored in the database. Subsequent requests for the same resource will include the stored time to avoid downloading resources that have not changed since previously obtained (and notionally still locally available). The uriLastModified.db files can be deleted with no penalty other than fresh retrieval of all remote artifacts associated with the database instance. The file should be deleted (or selectively edited) when automatically downloaded artifacts are manually deleted, else the artifacts may not be downloaded when needed due to a cached last modified time.



```
CREATE TABLE UriLastModified (
    UriId INTEGER PRIMARY KEY AUTOINCREMENT,
    Uri TEXT UNIQUE,
    LastModified TEXT);
```

## CRL Info

The crlIndex.db file is created to index CRLs present within a given folder. The index maps information useful in locating a CRL appropriate for determining the revocation status of a given certificate to a file name.

```
CREATE TABLE CrlFileInfo (
    CrlFileId INTEGER PRIMARY KEY AUTOINCREMENT,
    FullPathAndFilename TEXT,
    SKID TEXT,
    ThisUpdate BLOB,
    NextUpdate BLOB OPTIONAL,
    IssuerName TEXT,
    IssuerNameBlob BLOB,
    SigAlgBlob BLOB,
    CrlScope INTEGER,
    CrlCoverage INTEGER,
    CrlAuthority INTEGER,
    CrlReasons INTEGER,
    Extensions BLOB OPTIONAL,
    DistPointName TEXT OPTIONAL,
    DistPointNameBlob BLOB OPTIONAL,
    DeltaSeqNumber TEXT OPTIONAL
);
```

## Appendix C: Error Codes

The certificate validation dialog in the TACT Server Configuration Utility displays error codes reported by the certification path validation library. A description of each error code is below.

Error code name	Value	Description
NAME_CHAINING_FAILURE	25001	The subject name in a CA certificate or trust anchor does not match the issuer name in an immediately subordinate certificate in a certification path.
SIGNATURE_VERIFICATION_FAILURE	25002	The signature on a certificate could not be verified using the public key information from the immediately superior certificate in a certification path.
INVALID_NOT_BEFORE_DATE	25003	The not before value in a certificate in a

		certification path is later than the validation time of interest.
INVALID_NOT_AFTER_DATE	25004	The not after value in a certificate in a certification path is earlier than the validation time of interest.
MISSING_BASIC_CONSTRAINTS	25005	A purported CA certificate in a certification path is missing the basic constraints extension.
INVALID_BASIC_CONSTRAINTS	25006	A purported CA certificate in a certification path contains an invalid basic constraints extension value.
INVALID_PATH_LENGTH	25007	A certification path violated a path length constraint.
INVALID_KEY_USAGE	25008	A certificate contains an invalid key usage. For example, a CA certificate may not feature a key usage value that permits certificate signing.
NULL_POLICY_SET	25009	The valid policy tree became NULL during certification path validation.
NAME_CONSTRAINTS_VIOLATION	25010	A certificate in a certification path contains a name that violates a name constraint.
UNPROCESSED_CRITICAL_EXTENSION	25011	A certificate in a certification path contains a critical extension that was not processed.
CCC_UNAUTH_TA	25012	A CMS content constraints error was detected.
CCC_VIOLATION	25013	A CMS content constraints error was detected.
MISSING_TRUST_ANCHOR	25014	Certification path validation was attempted for a certification path that does not include a trust anchor.
MISSING_TRUST_ANCHOR_NAME	25015	A certification path includes a trust anchor that does not include a name and is thus not suitable for validating certification paths.
PROHIBITED_ALG	25016	A certificate in a certification path contains a public key or signature value associated with a prohibited algorithm.
PROHIBITED_KEY_SIZE	25017	A certificate in a certification path contains a public key that violates a key size constraint.
ENCODING_ERROR	25018	A BER/DER error was detected.
MISSING_CERTIFICATE	25019	A CMS SignedData message does not include a certificate required to validate the signature.
UNEXPECTED_CONTENT_TYPE	25020	A CMS message contains an unrecognized content type.
SEQ_NUM_VIOLATION	25021	A TAMP message contains a sequence number that is lower than the sequence number value stored for the message signer, i.e., the message is stale.
NO_PATHS_FOUND	25022	A certification path could not be found from a given end entity certificate to a trust anchor.
COUNTRY_CODE_VIOLATION	25023	The certification path includes a certificate that violates a country code constraint expressed in the indicated PathSettings.
CERTIFICATE_REVOKED	25024	The certification path includes a certificate that has been revoked.
REVOCATION_STATUS_NOT_DETERMINED	25025	The certification path contains one or more certificates for which revocation status could not be determined.
CERTIFICATE_ON_HOLD	25026	The certification path contains a certificate that has been placed on hold.

CERTIFICATE_BLACKLISTED	25027	The certification path includes a certificate that has been blacklisted in the specified revocation configuration.
STATUS_CHECK_RELIED_ON_STALE_CRL	25028	Revocation status checks utilized at least one stale CRL.
REVOCATION_STATUS_NOT_AVAILABLE	25029	No revocation status information was found for the certificate being checked.

## Appendix D: Sample Log Configuration

The following logging configuration corresponds to the example in the [Edit logging configuration](#) section above.

```
log4j.additivity.RhUtils=false
log4j.additivity.TaStoreManager=false
log4j.additivity.TactCli=false
log4j.additivity.TactComplianceAssessment=false
log4j.additivity.TactServerConfig=false
log4j.appender.rhUtilsAppender=org.apache.log4j.RollingFileAppender
log4j.appender.rhUtilsAppender.append=false
log4j.appender.rhUtilsAppender.fileName=d:\rhUtilsAppender.txt
log4j.appender.rhUtilsAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.rhUtilsAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.appender.rhUtilsAppender.maxBackupIndex=5
log4j.appender.rhUtilsAppender.maxFileSize=5000000
log4j.appender.rootCategoryAppender=org.apache.log4j.RollingFileAppender
log4j.appender.rootCategoryAppender.append=false
log4j.appender.rootCategoryAppender.fileName=d:\rollingFileAppender.txt
log4j.appender.rootCategoryAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.rootCategoryAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.appender.rootCategoryAppender.maxBackupIndex=5
log4j.appender.rootCategoryAppender.maxFileSize=5000000
log4j.appender.taStoreManagerAppender=org.apache.log4j.FileAppender
log4j.appender.taStoreManagerAppender.append=true
log4j.appender.taStoreManagerAppender.fileName=d:\taStoreManagerAppender.txt
log4j.appender.taStoreManagerAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.taStoreManagerAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.appender.tactCliAppender=org.apache.log4j.FileAppender
log4j.appender.tactCliAppender.append=true
log4j.appender.tactCliAppender.fileName=d:\tactCliAppender.txt
log4j.appender.tactCliAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.tactCliAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.appender.tactCliConsoleAppender=org.apache.log4j.ConsoleAppender
log4j.appender.tactCliConsoleAppender.layout=org.apache.log4j.SimpleLayout
log4j.appender.tactComplianceAssessmentAppender=org.apache.log4j.FileAppender
log4j.appender.tactComplianceAssessmentAppender.append=true
log4j.appender.tactComplianceAssessmentAppender.fileName=d:\tactComplianceAssessmentAppender.txt
log4j.appender.tactComplianceAssessmentAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.tactComplianceAssessmentAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.appender.tactServerConfigAppender=org.apache.log4j.FileAppender
log4j.appender.tactServerConfigAppender.append=true
log4j.appender.tactServerConfigAppender.fileName=d:\tactServerConfigAppender.txt
log4j.appender.tactServerConfigAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.tactServerConfigAppender.layout.ConversionPattern=%c - %d - %p - %m %n
log4j.category.RhUtils=ERROR,rhUtilsAppender
log4j.category.TaStoreManager=ERROR,taStoreManagerAppender
log4j.category.TactCli=ERROR,tactCliAppender
log4j.category.TactComplianceAssessment=ERROR,tactComplianceAssessmentAppender
log4j.category.TactServerConfig=ERROR,tactServerConfigAppender
log4j.rootCategory=ERROR,rootCategoryAppender
```



## Appendix E: Definitions

Path settings	Configuration item that includes: <ul style="list-style-type: none"> <li>• RFC 5280 certification path validation input definitions</li> <li>• RFC 5937 certification path validation input definitions</li> <li>• Cryptographic algorithm prohibition and cryptographic key size constraints enforcement indicator</li> </ul>
Security environment	Configuration item that defines: <ul style="list-style-type: none"> <li>• TACT trust anchor store to use during certification path validation</li> <li>• Certificate policies database to use during path settings configuration and logging of certification path validation results</li> <li>• Cryptographic algorithm prohibitions</li> <li>• Cryptographic key size constraints</li> </ul>
TACT	Trust Anchor Constraints Tool
TACT administrator	Prepares TACT settings for use in defining TACT resources using the <a href="#">TACT Server Configuration Utility</a>
TACT operator	Web server administrator who uses a TACT settings database with path settings and security environment definitions provided by a TACT administrator to define TACT resources in the <a href="#">TACT Server Configuration Utility</a>
TACT resource	A server resource subject to a path settings configuration and a security environment configuration. Expressed as a URI path or a file system path, for example: <ul style="list-style-type: none"> <li>• /</li> <li>• C:\inetpub\wwwroot</li> <li>• /var/www/html</li> </ul>
TACT settings	A collection of path settings and security environment configurations that may be used to define TACT resources
TASM	TA Store Manager utility
TA store administrator	Prepares and maintains trust anchor stores using the <a href="#">TA Store Manager</a> utility. Uses signed TAMP messages for remote management.
TCA	TACT Compliance Assessment utility
Trust anchor	An RFC 5914 TrustAnchorChoice containing a TrustAnchorInfo structure that wraps an X.509 certificate

TSC	TACT Server Configuration utility
-----	-----------------------------------

## Appendix F: Bibliography

RFC 5280	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
RFC 5652	Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.
RFC 5914	Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, June 2010.
RFC 5934	Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", RFC 5934, August 2010.
RFC 5937	Ashmore, S., and C. Wallace, "Using Trust Anchor Constraints during Certification Path Processing", <a href="#">RFC 5937</a> , August 2010.

## Appendix G: Sample Scripts

### Importing and exporting certificate policy information

The following sample Python script enables certificate policy information to be defined using a Comma-Separated Values (CSV) file and imported into a TACT certificate policies database or for information in a TACT certificate policies database to be exported to a CSV file.

```
import sqlite3
import csv
import logging
from optparse import OptionParser, OptionError

# getPolicyOid takes an object identifier in dot notation form (i.e., 1.2.3.4.5) and returns
# the integer ID of the OID from the database, if any. If the OID is not found, -1 is returned.
def getPolicyId(conn, oid):
    c = conn.cursor()
    query = "select CertPolicyId from CertificatePolicies WHERE PolicyOid=" + oid + ";"
    c.execute(query)
    id = -1
    for row in c:
        id = row[0]
    c.close()
    return id

# getPolicyOid takes the ID of a policy in the database and returns the object identifier.
def getPolicyOid(conn, id):
    c = conn.cursor()
    query = "select PolicyOid from CertificatePolicies WHERE CertPolicyId=" + str(id) + ";"
    c.execute(query)
    oid = ""
    for row in c:
        oid = row[0]
    c.close()
    return oid

# addPolicy takes an object identifier, a indication of association with the relying party domain
# and a descriptive name and adds a new item to the database.
def addPolicy(conn, oid, irpd, name):
    c = conn.cursor()
    query = "INSERT INTO CertificatePolicies VALUES(NULL, " + oid + ", " + irpd + ", " + name + ");"
    print query
    c.execute(query)
    conn.commit()
    c.close()

# addMapping takes a pair of database identifiers that identify policies present in the
# CertificatePolicies tables and creates a new entry in the PolicyMappings table where id
# is the IssuerDomain and mappedId is the subject domain,
def addMapping(conn, id, mappedId):
    print "addMapping " + str(id) + " " + str(mappedId)
    c = conn.cursor()
    query = "INSERT INTO PolicyMappings VALUES(" + str(id) + ", " + str(mappedId) + ");"
    print query
    c.execute(query)
    conn.commit()
    c.close()

# getMappings takes an object identifier in dot notation form (i.e., 1.2.3.4.5) and returns
# a string containing a semi-colon delimited list of object identifiers the provided OID
# maps to.
```

```
def getMappings(conn, oid):
    mappings = ""
    id = getPolicyId(conn, oid)
    if(-1 == id):
        return mappings;

    first = True
    c = conn.cursor()
    c.execute("select SubjectDomain from PolicyMappings WHERE IssuerDomain=" + str(id) + ";")
    for row in c:
        oid = getPolicyOid(conn, row[0])
        if(first):
            mappings = oid
            first = False
        else:
            mappings += ";" + oid
    c.close()
    return mappings
```

# addMappings takes an object identifier and a string containing a semi-colon delimited list  
# of object identifiers. It creates an entry in the PolicyMappings table for each element  
# in the delimited list where oid is the IssuerDomain and each element is a SubjectDomain  
# policy.

```
def addMappings(conn, oid, mappings):
    print "addMappings " + oid + " " + mappings
    id = getPolicyId(conn, oid)
    if(-1 == id):
        return
    for item in mappings.split(";"):
        mappedId = getPolicyId(conn, item)
        if(-1 != mappedId):
            addMapping(conn, id, mappedId)
```

# exportToCsv exports policies from the database identified by the --database option and writes  
# the information in comma-delimited form to the file identified by the --csv option.

```
def exportToCsv(options):
    print "exportToCsv"
    writer = csv.writer(open(options.csv, 'wb+'), delimiter=',')
    conn = sqlite3.connect(options.database)
    c = conn.cursor()
    c.execute("select PolicyOid, InRelyingPartyDomain, PolicyName from CertificatePolicies;")
    for row in c:
        mappings = getMappings(conn, row[0])
        writer.writerow([row[0], row[1], row[2], mappings])
    c.close()
```

# importFromCsv imports policies into the database identified by the --database option from a  
# comma-delimited CSV file identified by the --csv option.

```
def importFromCsv(options):
    print "importFromCsv"
    conn = sqlite3.connect(options.database)
    for row in reader:
        oid = row[0]
        irpd = row[1]
        name = row[2]
        addPolicy(conn, oid, irpd, name)
    reader = csv.reader(open(options.csv, 'rU'), delimiter=',')
    for row in reader:
        oid = row[0]
        mappings = row[3]
        addMappings(conn, oid, mappings)
```

```
def main():
    logger = logging.getLogger("")
    logger.setLevel(logging.INFO)
    handler = logging.StreamHandler()
    handler.setLevel(logging.DEBUG)
```



```

formatter = logging.Formatter("%(asctime)s [%(levelname)s] %(message)s")
handler.setFormatter(formatter)
logger.addHandler(handler)

usage = "Usage: %prog [options]"
parser = OptionParser(usage)
parser.add_option("-e", "--exporting", dest="exporting",
    help="when true data is exported from database to CSV, vice versa when false")
parser.add_option("-d", "--database", dest="database",
    help="file containing certificate policies database")
parser.add_option("-c", "--csv", dest="csv",
    help="file containing CSV formatted certificate policy information")
try:
    (options, args) = parser.parse_args()
except OptionError, e:
    logger.error("Unable to parse command line options: %s" % e.msg)
    sys.exit(1)

print options
if(options.exporting == "True"):
    exportToCsv(options)
else:
    importFromCsv(options)

if __name__ == "__main__":
    main()

```

## Downloading and applying a TAMP update message

The following Windows PowerShell script demonstrates how to download a TAMP update message for processing with the TactCli utility.

```

function main()
{
    $clnt = new-object System.Net.WebClient
    $urlbase = "http://locker.redhoundsoftware.com:8080/"
    $filename = "tu.tur"
    $targetdir = "C:\TactSettings\"

    $url = $urlbase + $filename
    $outfile = $targetdir + $filename
    $stamp = ( Get-Date ( Get-Date ).AddDays(-1) -uformat %m%d%Y%H%M%S )
    if(Test-Path $outfile)
    {
        write-host 'moving existing ' $filename ' out of the way.'
        $backupfile = $targetdir + $filename + "." + $stamp + ".bak"
        rename-item $outfile $backupfile
    }
    write-host 'downloading ' $url
    $clnt.DownloadFile($url,$outfile)

    $TactCli="C:\Program Files\TACT\bin\TactCli.exe"
    $database = $targetdir + "tas\default.tas"
    $logconfig = $targetdir + "tclog.properties"
    & "$TactCli" --action update --tampUpdate $outfile --database $database -l $logconfig
}

main
exit 0

```

## Retrieving information from the PKI log

The text logs generated by the TACT plugins include the event ID associated with a particular access event. Additional information can be retrieved from the database given this event ID. The sample Python script below can be used to retrieve access information, optionally including the certification path.

```
import sys
import sqlite3
import logging
from argparse import ArgumentParser,ArgumentError

def DumpCert(certId, index, results):
    conn = sqlite3.connect(results.database)
    c = conn.cursor()
    query = "select Certificate, SubjectName from Certificates WHERE CertificateId = " + str(certId) + ";"
    c.execute(query)
    if 0 == c.rowcount:
        c.close()
        return

    for row in c:
        fileName = results.outputFolder + "Event" + str(results.eventID) + "-Cert" + str(index) + ".der"
        f = open(fileName, 'wb')
        f.write(row[0])
        print "*" + Certificate #" + str(index) + " - " + row[1]
    c.close()

def DumpCerts(pathId, certId, results):
    conn = sqlite3.connect(results.database)
    c = conn.cursor()
    query = "select Certificate,PathIndex from PartialCertificationPathMembers WHERE PartialCertificationPath = " + str(pathId)
    + " ORDER BY PathIndex DESC;"
    c.execute(query)
    print "Certificate information for event #" + str(results.eventID)
    index = 0
    for row in c:
        DumpCert(row[0], index, results)
        index = index + 1
    c.close()
    DumpCert(certId, index, results)

def main():
    logger = logging.getLogger("")
    logger.setLevel(logging.INFO)
    handler = logging.StreamHandler()
    handler.setLevel(logging.DEBUG)
    formatter = logging.Formatter("%(asctime)s [%(levelname)s] %(message)s")
    handler.setFormatter(formatter)
    logger.addHandler(handler)

    usage = "Usage: %prog -d -e [-o] [-i] [-h]"
    parser = ArgumentParser(usage)
    parser.add_argument("-e", "--eventID", action='store', dest="eventID", default=-1, help="identifies the event to retrieve")
    parser.add_argument("-d", "--database", action='store', dest="database", help="file containing PKI log database", type=str)
    parser.add_argument("-o", "--outputFolder", action='store', dest="outputFolder", help="folder to receive event information")
    parser.add_argument("-i", "--infoOnly", action="store_true", dest="infoOnly", default=False, help="retrieve event information only - do not return certificates")
    try:
        results = parser.parse_args()
    except ArgumentError,e:
        logger.error("Unable to parse command line options: %s" % e.msg)
        sys.exit(1)
```

```
if -1 == results.eventID:
    logger.error("No event ID was specified.")
    parser.print_help()
    sys.exit(1)

if None == results.database:
    logger.error("No database was specified.")
    parser.print_help()
    sys.exit(1)

if False == results.infoOnly and None == results.outputFolder:
    logger.error("No outputFolder was specified.")
    parser.print_help()
    sys.exit(1)

conn = sqlite3.connect(results.database)
c = conn.cursor()
query = "select * from AccessLog WHERE EventId = " + str(results.eventID) + ";"
c.execute(query)
pathId = -1
certId = -1

if 0 == c.rowcount:
    print "No event found with ID " + str(results.eventID)
    sys.exit(1)

print ""
print "Access information for event #" + str(results.eventID)
for row in c:
    print "** Time:\t\t\t" + row[1]
    print "** SourceIP:\t\t" + row[2]
    print "** SourcePort:\t\t" + str(row[3])
    print "** DestIP:\t\t\t" + row[4]
    print "** DestPort:\t\t" + str(row[5])
    print "** UrlAccessed:\t\t" + row[6]
    print "** AccessAllowed:\t" + str(row[7])
    print "** CertId:\t\t\t" + str(row[8])
    print "** PathId:\t\t\t" + str(row[9])
    pathId = row[9]
    certId = row[8]
c.close()

if False == results.infoOnly:
    print ""
    DumpCerts(pathId, certId, results)
    print ""

if __name__ == "__main__":
    main()
```

## Appendix H: TAMP Profile

TACT does not implement RFC 5934 and related specifications in full. This section describes TACT's usage of RFC 5914, RFC 5934, RFC 5937 and RFC 6010:

- TACT trust anchor stores do not contain an Apex trust anchor.
- TACT does not use community identifiers or targeted TAMP messages. All TAMP messages are targeted to all devices (i.e., `TargetIdentifier.allModules`).
- TACT uses the following 5 TAMP message types: TAMP status query, TAMP status response, trust anchor update, trust anchor update confirm and TAMP error.
- Within trust anchor update, TACT supports add and remove operations. Edit operations are accomplished with a pair of remove and add elements within a trust anchor update message.
- TACT can use TAMP status response messages in a manner not stated in the specification, i.e., as a means of initializing a new trust anchor store.
- CMS Contents Constraints are used only to express content type constraints for the support TAMP message content types. Attribute constraints are not supported.
- TACT does not maintain sequence number state with the TAMP signer's credential. Sequence numbers are the number of seconds since 1970 using the current UTC time.
- Trust anchors always use the `TrustAnchorInfo` option within a `TrustAnchorChoice` with each instance containing a wrapped X.509 certificate along with optionally present constraints.

## Appendix I: TAMP Usage Strategies

TACT can be used with or without distribution of TAMP messages. Where TAMP messages are distributed to manage existing trust anchor stores, there are two basic approaches that can be adopted using TACT:

- Generate and distribute TAMP update messages with incremental changes
- Generate and distribute TAMP update messages with full contents

The following subsections discuss how to implement each approach along with pros and cons.

### Using TAMP update messages with incremental changes

The easiest means of using TAMP messages from the TA store manager's perspective is to maintain a single authoritative trust anchor store that is periodically edited with the signed TAMP message resulting from each edit distributed to interested parties. Since the messages are signed, the message can be distributed over unsecure channels. This approach is well suited for environments where trust anchor changes are infrequent and where recipients have reliable access to TAMP message distribution points. Web server administrators can define a scheduled task or cron job to periodically retrieve TAMP update messages for processing using the TactCli utility.

#### How to

Prior to creating a new trust anchor store, collect the following information:

- Trust anchors to include in the trust anchor store
- TAMP message signer's certificate (and private key, if generating TAMP messages)
- Any constraints to be associated with each trust anchor, i.e., name constraints, policies, etc.

To create a new trust anchor store, launch the TASM utility. Select the **File->New trust anchor store** menu. When prompted to select TA store manager(s), browse to the TAMP message signer's certificate (multiple certificates can be selected if located in the same folder, else additional TAMP message signer certificates can be added after the database is created). After selecting the TAMP message signers, the database will be created and the trust anchors list will contain only the keys associated with the TAMP message signers. To add trust anchors, click the **Add new TA...** button. Multiple certificate or trust anchor definitions can be imported at once if located in the same folder. After all trust anchors have been added and any constraints defined, click the **Apply changes** button and save the trust anchor store. The trust anchor store file can be distributed or a TAMP status response generated using the **Tools->Generate trust**

**anchor status message** menu item can be distributed to bootstrap applications. Messages digests of the database or TAMP status message should be calculated using the **Tools->Hash file** menu item then made available to enable recipients to confirm no changes have been made.

When a trust anchor store edit is required, open the trust anchor store file and make the necessary edits using the **Add new TA...**, **Remove selected TA** or **Edit selected TA...** buttons (or the TAMP authorization context menu accessed by right clicking a trust anchor in the list). To generate a signed TAMP message, ensure the intended TAMP Signer is selected in the drop list then click the **Apply changes** button. Retrieve the newly generated TAMP message from the TAMP folder specified in the options dialog and distribute using the desired mechanisms.

When distributing incremental changes, it may be wise to also make available TAMP status response messages to allow web server administrators an easy means of creating new trust anchor store files fully synchronized with the intended contents. To do so, simply click the **Tools->Generate trust anchor status message** menu item after generating each trust anchor update message. As noted above, message digests should be calculated and made available to allow recipients to confirm no changes have been made. Alternatively, complete trust anchor store files can be used as synchronization means or to allow for comparison using the TactCli or TACT Compliance Assessment utilities.

## Pros

For the TA store manager, this approach is very straightforward. Trust anchors are added, removed and edited with a history of TAMP messages accumulated in the TAMP message folder specified in the TASM options dialog. The TAMP messages resulting from this approach are smaller than other approaches.

## Cons

Depending on how messages are distributed, web server administrators may have a difficult time remaining in sync if an update message is missed. If only the latest message is made available, a web server that missed an update message may need to obtain a new complete trust anchor store or TAMP status response message prior to resuming periodic downloads of TAMP update message. If a collection of update messages is made available, the web server administrator must take care to apply them in order, else some messages will be unusable due to anti-replay mechanisms built into the TAMP messages.

## Using TAMP update messages with full TA store contents

For web server administrators, periodically downloading a trust anchor update message with the entire current contents of the trust anchor store is easiest. This avoids concerns about becoming out-of-sync with the intended contents and avoids problems associated with anti-replay mechanisms.

### How to

When using TAMP update messages with full TA store contents the TA store manager creates a new empty trust anchor store each time a change is required. Prior to creating a new trust anchor store, collect the following information:

- Trust anchors to include in the trust anchor store
- TAMP message signer's certificate (and private key, if generating TAMP messages)
- Any constraints to be associated with each trust anchor, i.e., name constraints, policies, etc.
- Any trust anchors that have been distributed in the past but should no longer be used by applications

To create a new trust anchor store, launch the TASM utility. Select the **File->New trust anchor store** menu. When prompted to select TA store manager(s), click cancel. An empty database will be created and the trust anchors list will be empty. First add the TAMP signers. Click the **Add New TA...** button and browse to the TAMP message signer's certificate (multiple certificates can be selected if located in the same folder, else additional TAMP message signer certificates can be added after the database is created). Next, right click each TAMP signer in the trust anchors list and choose **Add TAMP Authorization** from the context menu.

Next, add any trust anchors that have been used in the past but must not be used now. Click the **Add new TA...** button and browse to the desired trust anchors. The trust anchors list will now include TAMP message signers and trust anchors that should no longer be used. Select each trust anchor that should not be used and click the **Remove selected TA** button.

Next add trust anchors that should be used, click the **Add new TA...** button. Multiple certificate or trust anchor definitions can be imported at once if located in the same folder. Edit trust anchor definitions to define any constraints, as necessary, using the **Edit selected TA...** button. After all trust anchors have been added and any constraints defined, click the **Apply changes** button and save the trust anchor store. The trust anchor store file can be distributed.

Saving each trust anchor store as an archived edition following generation of a TAMP update message is recommended.

**Pros**

Distributing TAMP update messages with full contents is the easiest way to make sure there are no synchronization issues.

**Cons**

TAMP update messages with full contents are larger than incremental TAMP messages but the size should generally be well under 1MB. The steps required to generate a TAMP update message with full contents are somewhat more cumbersome for the TA store administrator, but it is a relatively infrequent task.



## Appendix J: Testing TACT settings with TSC

The TACT Server Configuration (TSC) utility provides tools to help test TACT settings and trust anchor stores prior to deployment. In addition to the information required to define the intended configurations, the following materials are required to perform tests using the **Tools->Validate Certificate** menu item in TSC:

- A set of one or more end entity certificates representing end entities in the communities to be serviced by a TACT installation.
- Intermediate certificates required for validation of each end entity certificate back to a trust anchor in a TACT trust anchor store.

End entity certificates are typically not published and must be obtained from individuals within each community of interest. Intermediate certificates are published and can be harvested. However, TACT does not perform any remote artifact collection so additional tools or resources are required. For testing with DOD approved external PKIs, a set of certificate is available from the DOD PKE site: <http://iase.disa.mil/pki-pke/interoperability/index.html>. For testing that requires other resources, the PKI Interoperability Test Tool (PITT) can be used to retrieve certificates from remote sources and saved locally for use with TSC. PITT is available here: <http://pkif.sourceforge.net/pitt.html>.

After collecting the resources to use during testing and defining the TA stores and TACT settings to test, launch TSC, navigate to the desired TACT settings database then select the **Tools->Validate Certificate** menu item. A dialog similar to the one shown below will be displayed.

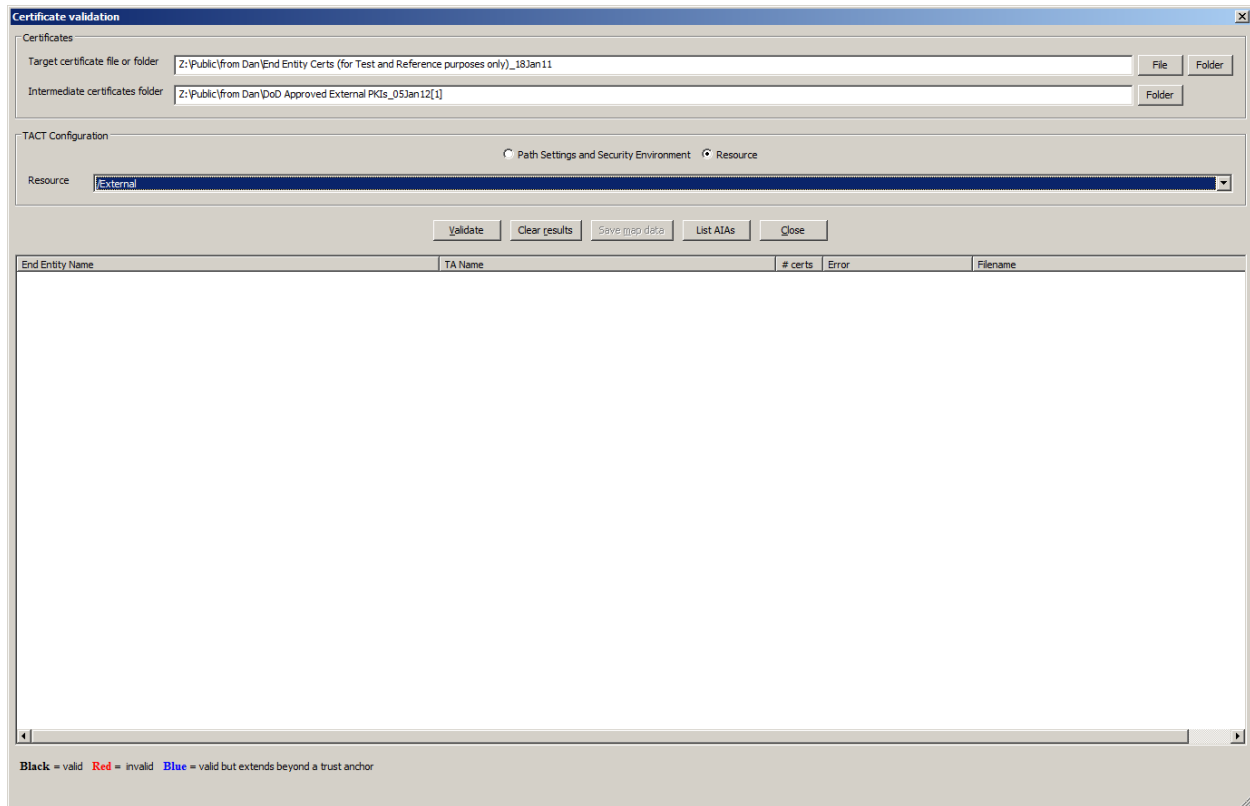


Figure 48 TSC certificate validation dialog

The **Target certificate file or folder** field is used to identify the certificate or collection of certificates that should be validated. To browse to a single certificate, click the **File** button adjacent to the field. To browse to a folder containing a collection of certificates click the **Folder** button adjacent to the field. Folders are recursively processed, so the certificate set can be organized into sub-folders beneath the folder identified here.

The **Intermediate certificates folder** is used to identify the collection of CA certificates to use when building paths from end entity (or target) certificates to a trust anchor in the trust anchor store identified by the selected security environment. As with target certificate folders, the intermediate certificates folder is processed recursively.

The **Path Settings and Security Environment** and **Resource** radio buttons determine whether the settings to test are selected by a pair of path settings and security environment configurations or by selecting a resource (that are associated with path settings and security environment configurations).

The **Path settings** and **Security environment** drop lists contain the names of the path settings and security environments included in the currently selected TACT settings database. The **Edit** buttons adjacent to each drop list can be used to edit the selected setting. Changes made using this edit capability are saved to the TACT settings

database, enabling iterative editing and testing from within the certificate validation dialog. The **Resource** drop list contains the names of the resources included in the currently selected TACT settings database. This option does not allow editing, but may be a more intuitive way for some server administrators to test TACT settings.

The **Validate** button is used to initiate a certification path build and validate operation. When **Validate** is clicked, TSC will load all certificates from the intermediate certificates folder and find all possible paths from each target certificate through the set of intermediate CAs to the trust anchors contained in the trust anchor store identified in the security environment and validate the path using the selected path settings. Different path settings and security environments can be defined to allow quick testing of slightly different configurations. Different path settings and security environments can also be defined to allow for tightly focused testing, for example, a security environment may be defined to reference a trust anchor store that contains a single trust anchor instead of a trust anchor store intended for operational use. The dialog below shows the results of validating a certificate using a set of intermediate CA certificates collected using PITT. In the list of results, lines colored red indicate an invalid path was found. Blue lines indicate a valid path was found but the path includes a public key that is a trust anchor (so the path is probably longer than necessary). Black lines indicate valid paths. In the diagram below, each blue path builds through the External Certification Authority (ECA) Root CA 2 key that terminates the single black line.

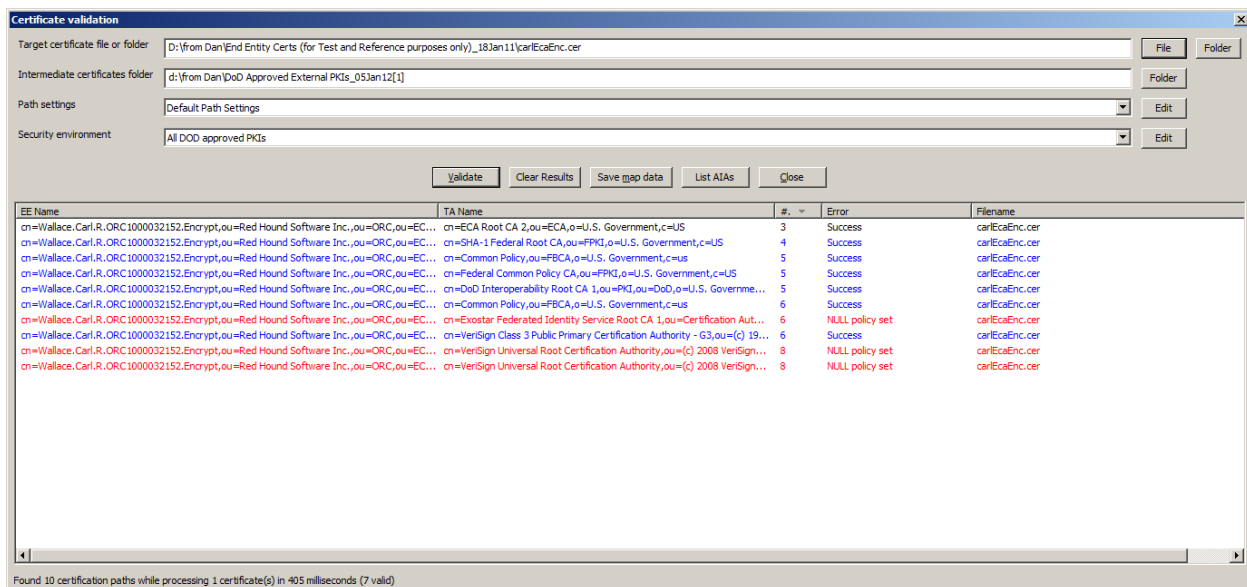


Figure 49 Certificate validation dialog with results

The **Clear Results** button can be used to clear the list of results. The **Close** button dismisses the certificate validation dialog.

The **Save Map Data** button can be used to save a DOT file containing a description of the PKI graph defined by the selected target certificate, intermediate certificates folder and trust anchor store. DOT files are plain text files that describe a graph and can be consumed by utilities like Graphviz to view an image of the graph. Graphviz is available here: <http://www.graphviz.org/>. After clicking **Save Map Data**, browse to a location where the DOT file should be saved.

The **List AIAs** button can be used to generate a list of all URIs from the authorityInfoAccess extensions in the certificates that comprise the paths displayed in the list. The List AIAs results figure below provides an example.

Details of items displayed in the list can be viewed by double clicking the item or right clicking the item and selecting **View results** from the context menu. The figure below shows an example results dialog.

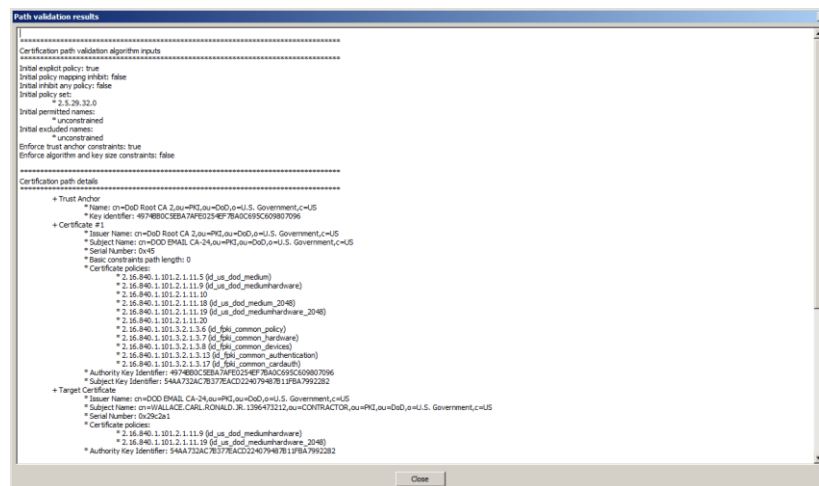


Figure 50 Path validation results

Certificates comprising the path can be saved by right clicking and selecting **Save results** from the context menu. The **List authorityInfoAccess URIs** item in the context menu can be used to collect a list of URIs from the authorityInfoAccess extensions in the certificates that comprise the path. This list can be useful in monitoring the remote resources for changes in the structure of the PKI associated with a TACT installation. The figure below shows an example results dialog following an AIA harvest operation.

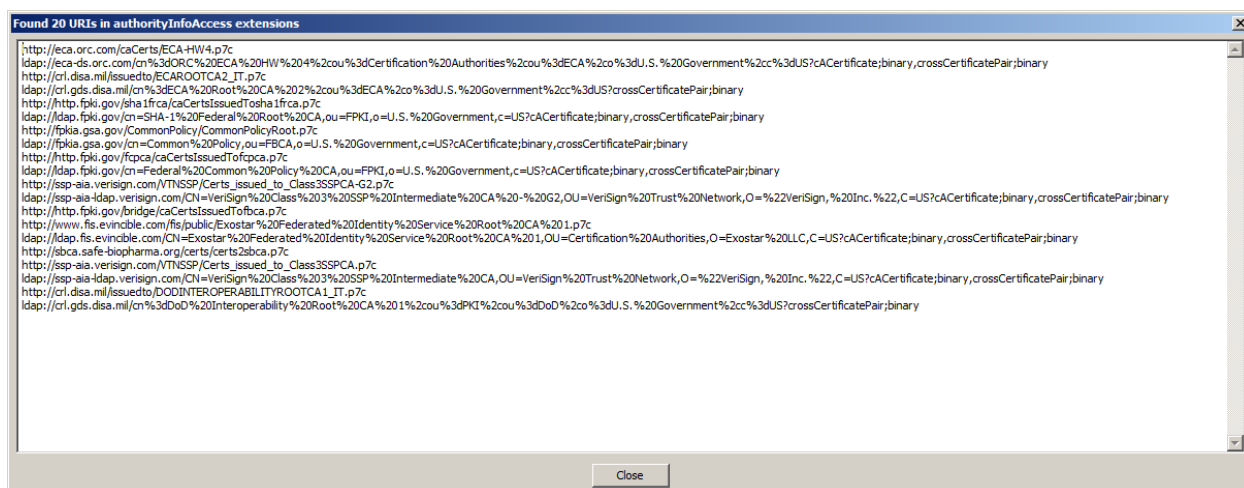


Figure 51 List AIAs results

## Appendix K: Default TACT Configuration Files

The TACT installer packages include the following nine configuration files:

- All\_DoD\_Approved.tas
- DoD\_ECA\_Federal.tas
- DoD\_ECA.tas
- DoD\_only.tas
- FedPolicies.pdb
- All\_DoD\_Approved.sdb
- DoD\_ECA\_Federal.sdb
- DoD\_ECA.sdb
- DoD.sdb

The All\_DoD\_Approved.tas file contains all trust anchors from the DOD approved PKIs, as noted here: <http://iase.disa.mil/pki-pke/interoperability/index.html>. The DoD\_ECA\_Federal.tas file is a reduction of the complete set that includes the DOD root, the DOD Interoperability Roots, the ECA root and all approved Federal PKI roots. The DoD\_ECA.tas file further reduces the set to include only the DOD root and the ECA root. The DoD\_only.tas file contains only the DOD root. Each file also includes a DOD trust anchor manager. Each trust anchor other than the DOD root and interoperability roots has an associated excluded name constraint for the DOD DN namespace.

The FedPolicies.pdb file is a certificate policies database containing selected policies from the set of DOD approved PKIs and includes mappings from DOD policies to these policies, where applicable. Federal PKI policies and associated mappings are also included.

Each of the .sdb files references contains four security environment definitions, a single path settings definition and a single TACT resource definition for /. The path settings definition turns on the required explicit policy flag and the enforce trust anchor constraints flags. Each security environment definition references a single trust anchor store (consistent with the name of the security environment) and FedPolicies.pdb. The TACT resource definition varies with the .sdb file. In All\_DoD\_Approved.sdb, the / resource references the security environment that includes All\_DoD\_Approved.tas. In DoD\_ECA\_Federal.sdb, the / resource references the security environment that includes DoD\_ECA\_Federal.tas. In DoD\_ECA.sdb, the / resource references the security environment that includes DoD\_ECA.tas. In DoD.sdb, the / resource references the security environment that includes DoD\_only.tas

The TACT installers use the All\_DoD\_Approved.sdb file by default. TACT operators are free to define additional resources that target alternative security environments or to change or remove the restrictions on /.

These files will be updated as new PKIs are approved or PKIs that are currently approved are decommissioned or become no longer approved. Updates will be available on the PKE interoperability site referenced above.

## Appendix L: Acronyms

<b>ACL</b>	Access Control List
<b>AIA</b>	Authority Information Access
<b>CA</b>	Certificate Authority
<b>CAPI</b>	Cryptographic Application Programming Interface
<b>CMS</b>	Cryptographic Message Syntax
<b>CSV</b>	Comma-Separated Values
<b>DNS</b>	Domain Name System
<b>DOD</b>	Department of Defense
<b>DOS</b>	Department of State
<b>ECA</b>	External Certification Authority
<b>IIS</b>	Internet Information Services
<b>IRCA</b>	Interoperability Root Certification Authority
<b>PITT</b>	PKI Interoperability Test Tool
<b>PKE</b>	Public Key Enablement
<b>PKI</b>	Public Key Infrastructure
<b>RDN</b>	Relative Distinguished Name
<b>RFC</b>	Request for Comments (Internet Engineering Task Force publication)
<b>RSA</b>	Rivest Shamir Adleman
<b>SHA</b>	Secure Hash Algorithm
<b>TA</b>	Trust Anchor
<b>TACT</b>	Trust Anchor Constraint Tool
<b>TactCli</b>	TACT Compliance Assessment Utility
<b>TAMP</b>	Trust Anchor Management Protocol
<b>TASM</b>	Trust Anchor Store Manager
<b>TCA</b>	TACT Compliance Assessment
<b>TLS</b>	Transport Layer Security
<b>TSC</b>	TACT Server Configuration
<b>UPN</b>	User Principal Name
<b>URI</b>	Uniform Resource Identifier
<b>UTC</b>	Coordinated Universal Time
<b>XML</b>	Extensible Markup Language



## Appendix M: Backout Plan

The following information describing how to disable or remove TACT is repeated here from the TACT Installation Instructions.

### Linux

#### Disabling an Installed Plug-in

To temporarily deactivate the TACT plug-in on a Linux system, apache needs to be prevented from loading the plug-in when it starts. Note that once TACT is disabled, any Trust Anchor (TA) constraints established in its TA store will not be enforced until it is re-enabled and Apache is restarted. On a default apache installation, renaming tact.conf is sufficient. Custom installations may require other steps. Consult these systems' build documentation for further guidance.

Step	Explanation	Example
1.	Become the super-user.	\$ su
2.	Stop the http server.	# /sbin/service httpd stop
3.	Rename the configuration file so that apache will ignore it upon restart.	# mv /etc/httpd/conf.d/tact.conf /etc/httpd/conf.d/tact.conf.disabled
4.	Start the http server.	# /sbin/service httpd start

When it's time to re-enable the plug-in, simply rename tact.conf.disabled to tact.conf and restart the server again.

#### Removing the TACT Plug-in on Linux

To permanently remove the TACT plug-in, simply uninstall the rpm containing it.

```
# rpm -e tactplugin
```

This will not remove any of the configuration or log files you've created on the system. Those must be removed manually, either from the custom selected locations or from the default /etc/tact and /var/log/tact.

### Windows

#### Disabling an Installed Plug-in

For Windows IIS servers, it is currently recommended to uninstall the plug-in in order to disable it. All configuration data will be left intact and can be used upon reinstallation. For Windows Apache servers, remove the line containing the tact.conf "Include" directive from the httpd configuration file.

## Uninstalling TACT

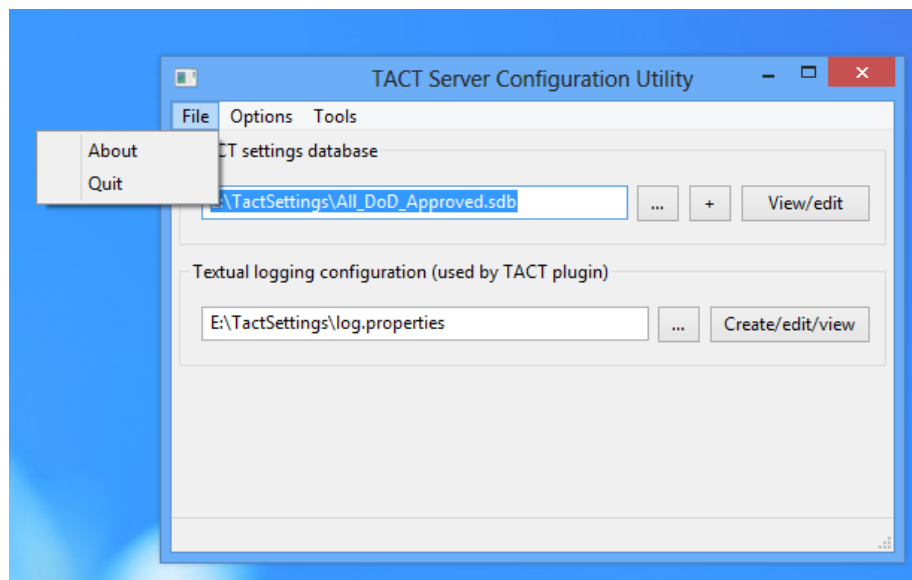
For Windows Apache servers, first remove the line containing the tact.conf “Include” directive from the httpd configuration file.

To uninstall TACT, use the Add/Remove Programs control panel. The uninstaller will not remove operator-created configuration data.

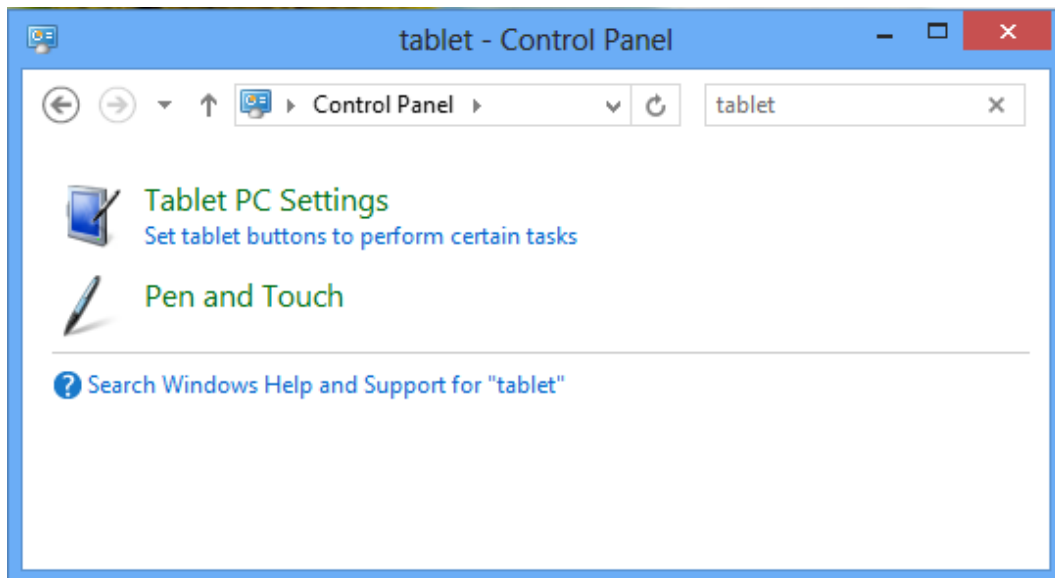
## Appendix N: Windows 8 GUI Appearance

Windows 8 introduced significant changes to the appearance of many graphical controls. It also attempts to enhance tablet/touchscreen behavior and expand tablet support to devices not typically considered “tablets,” even if these devices are not equipped with touch sensitive screens.

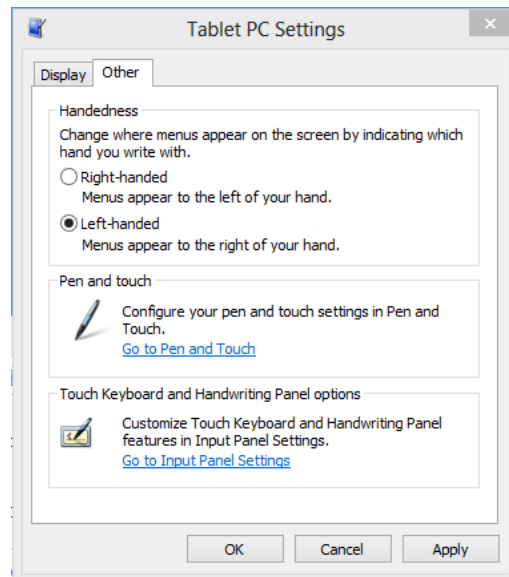
One of these enhancement attempts can cause certain menus to display in an unusual way in many desktop applications, including the TACT GUI administration utilities.



In order to restore the traditional menu appearance, the “Tablet PC Settings” control panel can be used. Open the control panel and use the search field on the right to search for “tablet.”



Within “Tablet PC Settings,” go to the “Other” tab and change “Handedness” to “Left-handed.”



Click apply. The menus in the TACT GUI tools will then be aligned normally, as will menus in other applications that are impacted by this setting.

## Appendix O: Reverting v1.2 settings files to v1.1 and earlier

TACT v1.2 automatically updates settings database files in a non-backward compatible way. Thus, files edited using TACT v1.2 utilities cannot be used by components from earlier TACT releases. While the earlier versions cannot be made to use new features in a TACT v1.2 settings database, old features can be edited using new tools with the resulting file being modified using the SQL statements below to allow older versions to use the file. To revert a settings database, use a sqlite utility (like the sqlite3 command line tool) to execute the following SQL commands.

```
DROP TABLE RevocationFreshnessConfigs;  
DROP TABLE LocalOcsp;  
DROP TABLE Whitelist;  
DROP TABLE Ignorelist;  
DROP TABLE Blacklist;  
DROP TABLE Certificates;  
PRAGMA user_version=1;
```

To ensure no v1.2 flags are set, rollback the database file using the above steps then open the database file with a v1.2 tool. This will update the version and turn off any new settings. Then repeat the SQL steps above.