Intel Cloud Integrity Technology 3.0

**Data Bundle**

# Background

When configuring web services, sometimes configuration or data needs to be imported from one service to another. For example, the Privacy CA AIK Issuer and the Attestation Server SAML Issuer certificates need to be imported to the Key Broker. The configuration bundle is a mechanism for the administrator to import such configurations in one step instead of one setting or object at a time.

In addition, sometimes it’s necessary to export and import data like users and permissions in order to persist across reinstalls. Although the size of data can be much larger than configuration, the problem is essentially the same.

# Architecture

## Data Export

Any service can provide downloadable configuration bundles covering different settings. For example one bundle might have all certificates. Another bundle might have the server configuration and TLS certificate. The service may let the user choose what is included in the bundle or it may generate predefined bundles.

The service MUST provide the bundle as a single file for the user to download and copy to the remote service.

## Data Import

Any service can provide a configuration bundle import function that accepts a single file, extracts configuration settings from it, and processes them.

The service MAY allow the user to upload more than one configuration bundle at a time.

## Bundle Format

The configuration bundle is a zip file that contains an index file and content. Each configuration setting is stored at a path which SHOULD be unique and predictable in order to allow services to interoperate using this format. The path should include the namespace of the feature. Within the feature’s directory, the settings or objects can be safely named arbitrarily.

The only files at the root directory of the archive should be support files for the data bundle feature itself such as the index file, and all other content and metadata should be stored in subdirectories with the namespace of the feature that specifies them.

## Extensions

In order to allow multiple features to export and import their settings within the same bundle, where these features are not known in advance, extension points are defined for providing information to export and for receiving information to import.

Importing settings and data is structured as a hook. Any component that needs to be notified about incoming bundles should implement the Contributor interface from mtwilson-core-data-bundle. The receive method will be called on all available Contributor implementations. Each Contributor receives the entire bundle as input and can search for any data or metadata that may be inside. Typically this is done by looking for files within a particular namespace.

## Security

Many settings are public such as certificates or URLs. However, some settings may be private such as the database username and password for a cluster, and even public settings may require integrity protection. Therefore, a configuration bundle may be encrypted with a password, secret key, or public key. In order to support a combination of unprotected, integrity protected, and encrypted content, the protections are applied within the bundle format and descriptor files are included to inform the importing service about how each part of the bundle is protected.

# Implementation

The core functionality is implemented in mtwilson-core-data-bundle. To add it as a dependency in Maven projects:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<classifier>feature</classifier>

<type>zip</type>

</dependency>

The feature provides packaging, export, import, and extraction capabilities.

For packaging it relies on TarGzipBuilder which is provided by:

<dependency>

<groupId>com.intel.mtwilson.util</groupId>

<artifactId>mtwilson-util-archive</artifactId>

<type>jar</type>

</dependency>

The difference between mtwilson-util-archive (generic utility) and mtwilson-core-data-bundle (mtwilson core feature) is that mtwilson-util-archive is a wrapper for commons-compress that attempts to simplify the use of the commons-compress API, while still being generic; whereas mtwilson-core-data-bundle makes assumptions about the structure of the archive itself, implements extensions, and includes a user interface plugin for mtwilson-core-html5.

## Status of implementation

This blueprint is not fully implemented. The attestation server and key broker sections below describe the current partial implementation.

Currently the data bundle does not include integrity or confidentiality security features. The security of this feature currently relies on the use of TLS to connect to the attestation server and the key broker, and on the user having control over the data bundle from the moment it is downloaded from the attestation server until the moment it is uploaded into the key broker.

## Data Bundle

The project features/mtwilson-core-data-bundle provides the TarGzipBundle class and an HTML5 interface for importing a bundle.

Currently the HTML5 interface defines the content tab for uploading the file but does not implement an extension for the configuration settings feature in order to appear dynamically there. This must be implemented when the settings feature is updated to support discovery.

## Configuration Settings

The project features/mtwilson-core-configuration-settings-ws-v2 includes a link under “More Settings” to upload a configuration data bundle.

Currently the link is embedded there and this must be converted to use discovery like the navbar does.

## Attestation Server

The project features/mtwilson-export-data-bundle creates a data bundle containing the SAML issuer and PCA AIK issuer certificates, the TLS certificate, and the attestation server URL, and provides an API to download this bundle. The java package is com.intel.mtwilson.configuration.data.bundle.

The following dependency was added to the project features/mtwilson-export-data-bundle:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<version>0.1-SNAPSHOT</version>

</dependency>

The following directory listing describes the content of the exported data bundle:

com.intel.mtwilson.configuration

+ EndorsementCA.pem

+ mtwilson.properties

+ PrivacyCA.pem

+ SAML.pem

+ TLS.pem

The JAX-RS endpoint is:

GET /v2/configuration/databundle

Content-Type: archive/tar+gz

The endpoint does not accept any parameters at this time.

The command line utility is:

mtwilson export-configuration-data-bundle [outfile]

The outfile parameter is the name of the file to which the bundle should be written.

NOTE: There is also a --stdout option to write the bundle content to stdout. At this time, it does not work because the mtwilson control script attempts to filter out stdout of all commands as a work-around for a log output issue, and piping the bundle through grep results in the message “Binary file (standard input) matches” output to stdout instead of the bundle content. When the control script supports sending output to stdout this feature will work and can be documented for users at that time.

The project portals/mtwilson-portal includes a new link “Download Configuration Data Bundle” in the certificate download page CertDownload.jsp which calls a corresponding function fnforConfigurationDataBundle() in CertDownload.js.

## Key Broker

The key broker relies on mtwilson-core-data-bundle to provide the user interface for importing a bundle, to extract the bundle and to call available extensions for processing its contents.

The following dependency was added to the project packages/kms-zip:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<classifier>feature</classifier>

<type>zip</type>

<version>0.1-SNAPSHOT</version>

</dependency>

The key broker project features/kms-saml defines extensions for processing the SAML issuer and PCA AIK issuer certificates in the data bundle.

The following dependency was added to the project features/kms-saml:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<version>0.1-SNAPSHOT</version>

</dependency>

The new classes DataBundleSamlCertificateImport and DataBundleTpmIdentityCertificateImport extend the abstract base class AbstractImportCertificatesPemToKeystore from mtwilson-core-data-bundle.

## Key Broker Proxy

The key broker proxy relies on mtwilson-core-data-bundle to provide the user interface for importing a bundle, to extract the bundle and to call available extensions for processing its contents.

The following dependency was added to the project packages/kmsproxy-zip:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<classifier>feature</classifier>

<type>zip</type>

<version>0.1-SNAPSHOT</version>

</dependency>

The key broker proxy project features-proxy/kmsproxy-saml-cache defines extensions for processing the TLS certificate and the attestation server URL in the data bundle.

The following dependency was added to the project features-proxy/kmsproxy-saml-cache:

<dependency>

<groupId>com.intel.mtwilson.core</groupId>

<artifactId>mtwilson-core-data-bundle</artifactId>

<version>0.1-SNAPSHOT</version>

</dependency>

The new class DataBundleTlsCertificateImport extends the abstract base class AbstractImportCertificatesPemToKeystore from mtwilson-core-data-bundle.

# Opens

## Integrity

Need to define a method to secure the integrity of any part of a data bundle.

Current proposal is to include a top-level “integrity” or “com.intel.mtwilson.core.data.bundle.integrity” folder which contains one or more JSON or PEM format files that describe the integrity checks for the bundle. Each such integrity file specifies the integrity algorithm, verification key, and manifest of one or more files to concatenate and digest for the integrity check.

An alternative is to provide an integrity check for the complete bundle but then it would be handled by an external component, which will force developers to add complexity to their application logic around data bundles and take away from the intent of the data bundle feature to provide a seamless import/export functionality for developers to integrate with security included.

When both confidentiality and integrity are included, the integrity component must run first.

## Confidentiality

Need to define a method to secure the confidentiality of any part of a data bundle.

Current proposal is to include a top-level “encryption” or “com.intel.mtwilson.core.data.bundle.encryption” folder which contains one or more JSON or PEM format files that describe the encryption settings for the bundle. Each such encryption file specifies the encryption algorithm, mode, padding mode, cipher key identity, and manifest of one or more files to encrypt (separately) using the same key.

When both confidentiality and integrity are included, the integrity component must run first.

## Permissions

Need to define a method to enforce the permissions associated with the bundle, especially when extensions (hooks) are used to receive the content. For example, a configuration data bundle exported from the attestation server and imported into the key server should have only configuration items in it - it should not include unrelated data like new users or modified key metadata.

Current proposal is the exporter may specify that the data bundle is intended for configuration only (by specifying the namespace, for example “com.intel.mtwilson.configuration”) and sign that assertion using the integrity feature, and the importer must check for such an assertion and if so, use the value provided to restrict what is provided to the extensions to only the specified namespace. That is, if additional data is added to the bundle in other namespaces, it would be hidden to any of the hooks that receive the bundle, and they will not be able to receive it.