Manage Configuration Drift

Customisable XML Baseline Guidance

The ‘Manage Configuration Drift’ Content Pack allows organisation to build customised baseline files which are constructed to contain areas of compliance settings seen as important for the operational integrity and compliance tracking of endpoints thus ensuring a good digital experience for the end user.

The configuration template itself is based on an .xml file, which can be configured to track the following areas of compliance.

- Windows Registry: Presence of Registry Keys, Registry Property Paths, Registry Key properties, Registry Key Values, and Registry Key Types
- Windows Services: Presence of Services (using short name), Service States, and Service Start-up types
- Files and Folders: Presence of files and folders on the device

Examples

Windows Registry

Check for the presence of registry keys on a device

Construct format

```
<!-- Define a list of registry keys that must be present on the device -->
        <RegistryKeyExistenceRequirements>
                <Key>HKEY_LOCAL_MACHINE\EXAMPLE_KEY</Key>
        </RegistryKeyExistenceRequirements>
```

Check for Registry Properties, Values, and Types

Construct format

```
<!-- Define a list of registry property paths, expected values and types to be checked on the device -->
        <RegistryPropertyWithExpectedValueAndTypeRequirements>
                <RegistryPropertyWithExpectedTypeAndValue>
                        <Key>HKEY_LOCAL_MACHINE\EXAMPLE_KEY</Key>
                            <Property>ExampleValue</Property>
                            <ExpectedValue>ValueExpected</ExpectedValue>
                            <ExpectedType>Example_such_as_REG_SZ</ExpectedType>
                </RegistryPropertyWithExpectedTypeAndValue>
        </RegistryPropertyWithExpectedValueAndTypeRequirements>
```

Note:
Only the following Windows Registry property types are accepted:

- REG_BIANRY
- REG_DWORD
- REG_EXPAND_SZ
- REG_MULTI_SZ
- REG_QWORD
- REG_SZ

Windows Services

The ability to check for the presence of Service presence on a device

Construct format

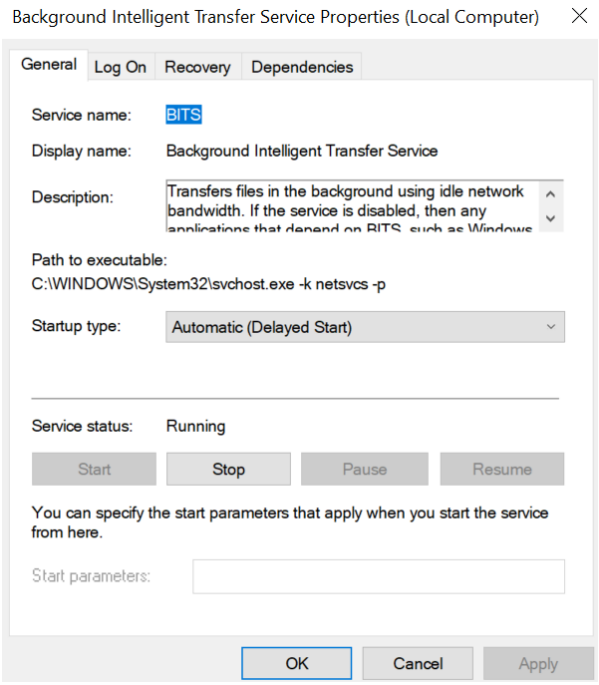
```
<!-- Define a list of service names that must be present on the device -->
        <ServiceExistenceRequirements>
                <ServiceName>Essential_Agent_Example</ServiceName>
                <ServiceName>Security_Service_Example</ServiceName>
        </ServiceExistenceRequirements>
```

The ability to check for the presence of Service States and Expected Start-up Types

Construct format

```
<!-- Define a list of services statuses and start types to be checked on the device -->
        <ServiceWithExpectedStartTypeAndStatus>
                <ServiceName>Example_Service_Name</ServiceName>
                <ExpectedStatus>Running</ExpectedStatus>
                <ExpectedStartType>Automatic</ExpectedStartType>
        </ServiceWithExpectedStartTypeAndStatus>
```

Note:
Windows Services should be entered in the format of 'Service Name'. In the case of "Background Intelligent Transfer Service" this would take the form of 'BITS' (as below)



Only the following Widows Services' statuses are accepted:

- Paused
- PausePending
- ContinuePending
- Running
- StopPending
- StartPending
- Stopped

Only the following Windows Services' types are accepted:

- Disabled
- Manual
- Automatic
- System
- Boot

Files and Directories

The ability to check for the presence of Files and Directories

Construct format

```
<!-- Define a list of files and directories that must be present on the device -->
        <FileDirectoryExistenceRequirements>
                <Path>C:\Program Files\Example Directory</Path>
                <Path>C:\Example_Critial_File\Definition-v-1.2.3.def</Path>
        </FileDirectoryExistenceRequirements>
```

The ability to check for the presence of Files with the expected checksum

Construct format

```
<!-- Define a list of files and checksums to be checked on the device -->
        <FileWithExpectedChecksumRequirements>
                <FileWithExpectedChecksum>
                <Path>c:\Example_Antivirus\definition.dat</Path>
                <ExpectedChecksum>12345665468484351681351684651354351351351651687987651685195814 65</ExpectedChecksum>
        </FileWithExpectedChecksum>
```

Note:

To extract the checksum of a file, run the following in PowerShell substituting [path to file] with the path to the appropriate file, ie C:\Windows\System32\kernel32.dll

```
Get-FileHash -Path [path to file] -Algorithm 'SHA256'
```