1 Introduction

1.1 Purpose and Scope

This document describes the newly restructured and redesigned SIS file format, which is introduced to complement the Symbian OS v9.1 release of Software Install. SIS files are used as the primary means of packaging files for deployment to a device, and are interpreted accordingly by the native software installer. As well as simply containing files, the SIS file can contain conditional statements which influence the installation e.g. device specific installations, language-specific installations, and user-selectable optional components.

Symbian OS v9.1 delivers new security features to the device, so operations which were previously possible via software install may now no-longer be possible. In addition, the device-side native installer is now policing the installation to ensure that the package meets certain security criteria before installation can succeed.

2 SIS File Format

2.1 Overview

The information in the SIS file is split up into two separate parts. The first part is the meta-data, describing the files that need to be installed. The second part of the SIS file contains all the actual file data. This enables software installation to be split into two phases, a decision and an installation phase. During the decision phase, the SIS file is examined and security checks are carried out in order to verify the install. The installation phase is only carried out if the verification is successful and is the process of copying the files to the device.

2.1.1 Note on Reservation of SIS Field Values

Please note that Symbian reserves the right to extend the set of applicable values relating to defined SIS fields, for example, fields employing bit-indicators such as TInstallFlags.

The generation of unspecified values – outside the context of the supported MakeSIS and/or SignSIS tools is therefore likely to cause compatibility problems with later versions of Symbian OS. This may mean, therefore, that packages will fail to install.

2.2 Integrity

The SIS file format supports signatures and certificates to enable a package to be signed. These signatures are verified during installation, and can also be re-verified after the package is installed on the device.

In order to support the processing of the SIS file in two phases, only the meta-data of the SIS file is signed. The metadata contains hashes for each file in the package, in order to ensure the integrity of the file data, and therefore the integrity of the entire SIS file is protected by the signed meta-data. This means that during the installation phase software install can verify the hash against the one included in the signed meta-data for each file being installed, whilst using an untrusted component to perform any necessary decompression.

Separate checksums for each of the meta-data and the file data are present in the SIS file to enable corrupt SIS files to be detected at the beginning of the installation process. These checksums are optional.
2.3 Extensibility
Due to the effort and potential disruption involved in changing file formats, the SIS format is designed to be extensible, and uses a type-length-value (TLV) format. Since each SISField has a specified length, when parsing a SIS file the installer will ignore fields with unknown types.

2.4 Compression
The new SIS file format supports the compression of each of the files in the SIS file individually, and the SISController can also be compressed. This reduces the extra space needed to carry out installation. Compression is supported by using a SISCompressed SISField which can contain another compressed SISField inside.

2.5 Nesting
In order to limit the amount of resource required to install a SIS file package, the installer will now only processes the nesting of SIS files down to a depth of 8. SIS files containing embedding to a greater depth than this will be rejected.

2.6 Supporting the re-signing of a SIS file
Since the SIS file format has no offsets which need to be changed it is easy to add a new signature and certificate chains to the end of the meta-data of the SIS file, even though they are in the middle of the file.

```
Symbian File Header
SISContents
.....
SISController
.....
.....
SISSignatures
SISData
.....
```

The SISSignatures SISField will be lengthened by the addition of additional signatures and certificate chains, and the SISFields following will be moved to a position further on in the file.

2.7 Programming Considerations

2.7.1 File Format
The SIS file format is designed so that each type of SISField is represented by one class. This makes it easy to construct a C++ class instance from a SISField. Since there are no offsets used in the file format it is possible to construct a C++ class with just the data from the SISField.
2.7.2 Memory usage
Since a SIS file may be large it is not possible to load everything into memory at once. Due to the structure of the file format, the meta-data information of each SISField can be read without reading all of the data in the contained SISFields.

2.8 Supporting the embedding of a SIS file
The SIS supports the embedding of one SIS file into another. MakeSIS is able to take an already generated SIS file and embed it into a SIS file that it is creating. The existing SIS file will be loaded, and the SISController decompressed if necessary and inserted into the Embedded SIS Files field of the SISInstallBlock. The SISDataUnit which contains the files needed for installation are added onto the end of the Data Units array of the SISDataSISField. Since the SISControllers have a Data Index field, which indicates the index of the SISDataUnit which contains the files they need, MakeSIS must iterate through the added SISControllers and change these to the correct values.

![Figure 1 – Embedding a SIS file](image)

In the resultant SIS file, to find the absolute index of the SISFileData in the SISDataUnit, the data indices of each of the chain of SISControllers, from the outermost SISController to the
SISController currently being considered, are summed. From Figure 1 – Embedding a SIS file, both of the SISControllers have a Data Index equal to zero. When calculating the index of the SISFileData in the SISDataUnit for Controller 2, the data indices of Controller 1 and Controller 2 are summed to get the absolute index of one.

![Diagram of SISControllers and Data Indices](image)

Indicates Embeds

**Figure 2 – Data indices with multiple embedded controllers**

The following table indicates the absolute index of the SISDataUnit in SISFileData, for the corresponding controller in the previous diagram.

<table>
<thead>
<tr>
<th>SISController</th>
<th>Absolute Index of SISDataUnit in SISFileData</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A (0) = 0</td>
</tr>
<tr>
<td>B</td>
<td>A (0) + B (1) = 1</td>
</tr>
<tr>
<td>C</td>
<td>A (0) + B (1) + C (1) = 2</td>
</tr>
<tr>
<td>D</td>
<td>A (0) + D (3) = 3</td>
</tr>
<tr>
<td>E</td>
<td>A (0) + D (3) + E (1) = 4</td>
</tr>
<tr>
<td>F</td>
<td>A (0) + D (3) + F (2) = 5</td>
</tr>
<tr>
<td>G</td>
<td>A (0) + G (6) = 6</td>
</tr>
<tr>
<td>H</td>
<td>A (0) + G (6) + H (1) = 7</td>
</tr>
</tbody>
</table>

**2.9 Byte ordering**

All meta-data are stored in little-endian format.
2.10 Text Character Set
The new SIS file format only supports Unicode UCS-2 encoded strings.

2.11 File Limitations
The number of levels of embedding of SIS controllers/files is limited to eight, as previously covered.

Various installation types have been removed. The supported types are now:

- EInstInstallation // standard type
- EInstAugmentation // removable addition to a package
- EInstPartialUpgrade // adds files to a package (patch) without any removals
- EInstPreinstalledApp // for use with pre-installed media
- EInstPreInstalledPatch // for use with pre-installed media

The unsupported types are EInstSISSystem, EInstSISOption, EInstSISConfig, EInstSISPatch, EInstSISMidlet and EInstSISMidletSuite.

2.12 SIS File Structure Overview
The SIS file format is composed of SISFields encoded using a type-length-value format. All SISFields are stored in this format, with the exception of any SISField which is stored inside a SISArray. This is since an array stores SISFields of the same type, it would be inefficient to store the type value for each entry in the array, and so only the Length and Values are stored.

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
</tr>
</tbody>
</table>

2.12.1.1 Type
This field indicates the type of the SISField. Each type of SISField has a unique ID, details of which can be found in Appendix A.

The type field is 4 bytes in length.

2.12.1.2 Length
This is the length of the Value field only, and does not include the sizes of the other fields contained in the SISField.

The Length field is stored in either 4 or 8 bytes, depending on its value. This is because for some fields we need to support a 64 bit length but for most we don’t, so storing the length in 64 bits for all fields would use unnecessary space. The Length is always represented by an unsigned value.

If Length is smaller than $2^{31}$ then the value is stored using 32 bits. If Length is greater than or equal to $2^{31}$ then the value is stored using 64 bits. The most significant bit is set to one, meaning the greatest possible Data Length which can be presented is $2^{63} - 1$.

To read in the value of Data Length we first read in the first 32 bits. If the most significant bit is zero, then the lower 31 bits represent the value of Length. If the most significant bit is one, then we read the next 32 bits, and construct the 63 bit data value from both parts.

2.12.1.3 Value
This field contains the data of the SISField. Its format depends on the Field ID.
2.12.1.4 Rationale
The reason for this format is that it makes it very easy to construct a C++ class instance from a SISField. It is also possible to construct this instance by giving only the SISField data and no other part of the SIS file.

2.12.2 Alignment
The SIS file is padded with zero bytes (at the end of each SISField) where necessary so each SISField begins on a 32 bit word boundary. This is to enable efficient parsing of the format from memory, on processors which only allow 32-bit aligned access.

2.12.3 Notation
The following notation is used to describe the data-structures used by the SIS file format:

<table>
<thead>
<tr>
<th>Structure Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of Field 1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Name of Field N</td>
</tr>
</tbody>
</table>

The Structure name is the name of the structure, which determines the ID stored in the type field. The length is determined by the length of all the fields specified. The fields 1 to N, specify the data which should appear in the value part of the structure.

3 SIS File Structure
All of the actual data of the SIS file is contained in the SISContents SISField. However Symbian OS uses a header, to associate files with applications. This header is a flat data structure consisting of three 32-bit UIDs and a 32-bit checksum.

<table>
<thead>
<tr>
<th>Symbian File Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISContents</td>
</tr>
</tbody>
</table>

3.1 File Header Structure
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID 1</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>UID 2</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>UID 3</td>
<td>4 Bytes</td>
</tr>
<tr>
<td>UID Checksum</td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>
3.1.1 UID 1
This is the UID of the application associated with SIS files. This is always 0x10201A7A.

3.1.2 UID 2
This UID is reserved for possible future use.

3.1.3 UID 3
This package UID identifies the SIS file. This UID will uniquely identify the SIS file, except for the case of upgrades, where both SIS files will share the same UID3 This UID should be the same as the UID present in the SISUid SISField in the top level SISController.

Note that this UID (UID3) is generally referred-to in other documentation as the package-UID, or pUID for short.

3.1.4 UID Checksum
This field provides a checksum of the UID structure in its entirety.

4 SIS Fields

4.1 General SISFields

4.1.1 SISString
This SISField contains a UCS-2 encoded Unicode string.

<table>
<thead>
<tr>
<th>SISString</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

4.1.1.1 String
This field contains the Unicode UCS-2 encoded string. Its length in bytes is specified by the Length field, and since each character is encoded using 16 bits there will be half as many characters in the string, as specified by this length.

4.1.2 SISArray
The SISArray SISField holds an array of one SISField type. The type of the contained SISFields will be checked on creation from data, and addition of each new SISField. The notation SISArray <SISString> will be used to indicate an array of SISStrings. All of the SISFields in the array are stored without their type as an optimisation, so just the length and value parts of the TLV are stored.

<table>
<thead>
<tr>
<th>SISArray</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISField Type</td>
<td>TUint32</td>
</tr>
<tr>
<td>SISField 1</td>
<td>SISField</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>SISField N</td>
<td>SISField</td>
</tr>
</tbody>
</table>
4.1.2.1 SISField Type
This field indicates the type of the SISFields in the array. All of the fields are of the same type and this will be checked on creation of the SISField from data, and addition of each new SISField.

4.1.2.2 SISFields
This is a sequence of SISFields, whose type is equal to the value of the SISField type field. The SISField is only partially stored, the type being left out, as an optimisation since it can be determined from the SISField Type field of the SISArray. The number of fields can be determined by reading in all the fields until we have read all the data specified by the Length of the SISArray SISField.

4.1.2.3 Rationale
In several places in the SIS file format an array of SISFields is needed, so in order to reduce code duplication a SISArray type is provided.

4.1.3 SISCompressed
This SISField is a wrapper around raw data, where the wrapped data can be optionally compressed. This data can be a SISField which allows easy integration of compression into the SIS file format. The notation SISCompressed<SISString> will be used to indicate a compressed SISString, and SISCompressed<Raw Data> for compressed raw data.

<table>
<thead>
<tr>
<th>SISCompressed</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression Algorithm</td>
<td>TUInt32</td>
</tr>
<tr>
<td>Uncompressed Data Size</td>
<td>TUInt64</td>
</tr>
<tr>
<td>Compressed Data</td>
<td></td>
</tr>
</tbody>
</table>

4.1.3.1 Compression Algorithm
This field contains the algorithm used to compress the data for this file.

```plaintext
e num TCompressionAlgorithm
{
    ECompressNone = 0,    // The data is uncompressed
    ECompressDeflate     // The data is compressed according to RFC 1951
};
```

4.1.3.2 Uncompressed Data Size
This field contains the size of the data, when it is uncompressed.

4.1.3.3 Compressed Data
This field contains the raw compressed data.

4.1.4 SISVersion
This SISField provides a data structure for the storage of a version number, with major, minor and build components.

<table>
<thead>
<tr>
<th>SISVersion</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>TUInt32</td>
</tr>
<tr>
<td>Minor</td>
<td>TUInt32</td>
</tr>
<tr>
<td>Build</td>
<td>TUInt32</td>
</tr>
</tbody>
</table>
Only positive values or zero can be used to indicate a specific version. However, where applicable, the Major, Minor, or Build components of the SISVersion can be set to –1 in order to indicate any version.

### 4.1.5 SISVersionRange

This SISField specifies a range of versions. It is used to indicate which versions can satisfy a certain dependency. If the range is only a specific version then both the ‘From Version’ and ‘To Version’ fields should be set to the same specific value. If the upgrade is applicable to any version then both the ‘From version’ and the ‘To Version’ should have the Major, Minor and Build components set to –1. The ‘To Version’ field may be omitted, meaning the version range applies to the ‘From Version’ and all subsequent versions.

<table>
<thead>
<tr>
<th>SISVersionRange</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Version</td>
<td>SISVersion</td>
</tr>
<tr>
<td>To Version</td>
<td>SISVersion</td>
</tr>
</tbody>
</table>

### 4.1.5.1 Version Checking

When checking a dependency, we check the installed version of the package against the ‘From Version’ and the ‘To Version’ separately.

To check the ‘From Version’ we first check that the major version, of the package being installed, against the major version of the installed version. If the installed major version is less then this dependency check fails. If the installed major version is greater than this dependency check passes. If they are equal then we check the minor version in the same way. If all the components of the versions are equal then the dependency check passes. In this way we carry out a lexicographical compare of the versions. The value of -1 in any of the major, minor or build versions is treated as a special case. If we reach a compare with a field where the ‘From Version’ is -1, then the whole of the from part of the dependency check passes.

The ‘To Version’ is checked in a similar way.

Examples:

<table>
<thead>
<tr>
<th>Major</th>
<th>Minor</th>
<th>Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Version</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>To version</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

This will upgrade any version from 3.x.x to 4.5.x where x is any value.

<table>
<thead>
<tr>
<th>Major</th>
<th>Minor</th>
<th>Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Version</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>To version</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

This will upgrade either 1.3.4 or 1.3.5.

### 4.1.6 SISDate

This SISField contains a date. The date is stored according to the Gregorian calendar, with the year part being stored in full, and must be a valid date.

<table>
<thead>
<tr>
<th>SISDate</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>TUint16</td>
</tr>
</tbody>
</table>

2 bytes
**Month**

| **TUint8** | 1 byte |

**Day**

| **TUint8** | 1 byte |

### 4.1.6.1 Year

The year is stored as an absolute number, i.e. the year 2004 is represented by storing 2004 in this field.

### 4.1.6.2 Month

Months are stored using 0 for January up to 11 for December.

### 4.1.6.3 Day

Days are stored beginning from one.

### 4.1.7 SISTime

This SISField contains a time. The time must be expressed in UTC, and be a valid time.

<table>
<thead>
<tr>
<th><strong>SISTime</strong></th>
<th><strong>Length</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours</td>
<td><strong>TUint8</strong></td>
</tr>
<tr>
<td>Minutes</td>
<td><strong>TUint8</strong></td>
</tr>
<tr>
<td>Seconds</td>
<td><strong>TUint8</strong></td>
</tr>
</tbody>
</table>

### 4.1.8 SISDateTime

This SISField contains both date and time SISFields.

<table>
<thead>
<tr>
<th><strong>SISDateTime</strong></th>
<th><strong>Length</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>SISDate</td>
</tr>
<tr>
<td>Time</td>
<td>SISTime</td>
</tr>
</tbody>
</table>

### 4.1.9 SISUid

This SISField contains the UID of the SIS file.

<table>
<thead>
<tr>
<th><strong>SISUid</strong></th>
<th><strong>Length</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>UID 1</td>
<td><strong>TInt32</strong></td>
</tr>
</tbody>
</table>

### 4.1.10 SISLanguage

This SISField identifies a language.

<table>
<thead>
<tr>
<th><strong>SISLanguage</strong></th>
<th><strong>Length</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td><strong>TUint32</strong></td>
</tr>
</tbody>
</table>

### 4.1.10.1 Language

The value of this field corresponds to the TLanguage enumeration, but is stored as a TUint32 in the SIS file.
4.1.11 SISBlob
This SISField encapsulates some raw data which is interpreted depending on the context.

<table>
<thead>
<tr>
<th>SISBlob</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

4.1.11.1 Data
This field contains the data. Its length is determined by the Length of the SISField.

4.1.12 SISDataIndex
This SISField is used as a relative index into the array of Data Units field of the SISData SISField. To get the absolute index of the SISDataUnit in the SISFileData SISField, all the data indices from the outermost embedding SISController down to the SISController currently being considered are summed. See Supporting the embedding of a SIS file for more information.

<table>
<thead>
<tr>
<th>SISDataIndex</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Index</td>
<td>TUint32</td>
</tr>
<tr>
<td></td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

4.1.12.1 Data Index
This field is used to find the index into the array of Data Units field of the SISData SISField. There is one SISDataUnit for each SISController.

4.2 SIS File Meta-data SISFields

4.2.1 SISContents
This SISField contains the whole of the contents of the SIS file. The contents are split up into the SISController, which contains the meta-data, and the SISData, which contains the actual file data.

<table>
<thead>
<tr>
<th>SISContents</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller Checksum</td>
<td>SISControllerChecksum</td>
</tr>
<tr>
<td>Data Checksum</td>
<td>SISDataChecksum</td>
</tr>
<tr>
<td>Controller</td>
<td>SISCompressed</td>
</tr>
<tr>
<td>Data</td>
<td>SISData</td>
</tr>
</tbody>
</table>

4.2.1.1 Controller Checksum
The checksum is a CRC-16 checksum over the contents of the Controller field. The checksum is over the whole of the SISCompressed<SISController>, so if the SISController is compressed, it does not have to be decompressed to verify this checksum. This enables the checking of the integrity of the controller without checking the whole file. This field is optional, and may not be present.

4.2.1.2 Data Checksum
The checksum is a CRC-16 checksum over the contents of the Data field. This enables the checking of the integrity of the data without checking the whole file. This field is optional, and may not be present.

4.2.1.3 Controller
The controller contains all the meta-data for the SIS file.
### 4.2.1.4 Data

The data field contains the actual files in the SIS file. These are processed differently depending on the meta-data present in the controller field.

### 4.2.2 SISControllerChecksum

This **SISField** contains the checksum for the possibly compressed **SISController**.

<table>
<thead>
<tr>
<th>SISControllerChecksum</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checksum</td>
<td>TUint16</td>
</tr>
</tbody>
</table>

#### 4.2.2.1 Checksum

This field contains the CRC-16 checksum, which is calculated over the whole of the **SISCompressed<SISController>SISField**.

### 4.2.3 SISDataChecksum

This **SISField** contains the checksum for the **SISData** section of the SIS File.

<table>
<thead>
<tr>
<th>SISDataChecksum</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checksum</td>
<td>TUint16</td>
</tr>
</tbody>
</table>

#### 4.2.3.1 Checksum

This field contains the CRC-16 checksum, which is calculated over the whole of the **SISDataSISField**.

### 4.2.4 SISController

This **SISField** contains the meta-data for the SIS file.

<table>
<thead>
<tr>
<th>SISController</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>SISInfo</td>
</tr>
<tr>
<td>Options</td>
<td>SISSupportedOptions</td>
</tr>
<tr>
<td>Languages</td>
<td>SISSupportedLanguages</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>SISPrerequisites</td>
</tr>
<tr>
<td>Properties</td>
<td>SISProperties</td>
</tr>
<tr>
<td>Logo</td>
<td>SISLogo</td>
</tr>
<tr>
<td>Install Block</td>
<td>SISInstallBlock</td>
</tr>
<tr>
<td>Signature 0..N 0</td>
<td>SISSignatureCertificateChain</td>
</tr>
<tr>
<td>Signature 0..N ...</td>
<td>SISSignatureCertificateChain</td>
</tr>
<tr>
<td>Signature 0..N N</td>
<td>SISSignatureCertificateChain</td>
</tr>
<tr>
<td>Data Index</td>
<td>SISDataIndex</td>
</tr>
</tbody>
</table>
4.2.4.1 Info
This field contains information about the SIS file.

4.2.4.2 Options
This field contains the options that the user is asked to choose from when installing the file. These options are used to determine which files to install.

4.2.4.3 Languages
This field contains the languages supported by the SIS file.

4.2.4.4 Prerequisites
This field contains the prerequisites needed in order to install the SIS file.

4.2.4.5 Properties
This field contains properties, which are key, value pairs of integers.

4.2.4.6 Logo
This field is optional, and if present contains a logo which will be displayed at the start of installation.

4.2.4.7 Signature 0..N
These SI SFields contain signatures, which sign the data contained in the SISController, excluding the Data Index SISField. Each SI SSignatureCertificateChain SISField signs the data from the beginning of the SI SInfo SISField in the SISController to the end of the SI SField immediately preceding the SI SSignatureCertificateChain SISField. Thus, each signature signs all the previous signatures as well as the SISController data.

There may be any number of SI SSignatureCertificateChain SISFields, including zero, meaning the SISController is unsigned.

4.2.4.8 Data Index
This SI SField is an index into the array of Data Units field of the SI SData SI SField. There is one SI SDataUnit for each SI SController.

4.2.5 SISInfo
This SI SField contains information about the SIS file.

<table>
<thead>
<tr>
<th>SISInfo</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>SISUid</td>
</tr>
<tr>
<td>Vendor Name</td>
<td>Unique Name</td>
</tr>
<tr>
<td>Names</td>
<td></td>
</tr>
<tr>
<td>Vendor Names</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Creation Time</td>
<td></td>
</tr>
<tr>
<td>Install Type</td>
<td></td>
</tr>
<tr>
<td>Install Flags</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2.5.1 UID
This field contains the UID of the SIS file. The UID should be unique to a SIS file packaging a certain application, but there may be multiple different versions of this package, with the same UID.

4.2.5.2 Vendor ID
This field contains the ID of the vendor which created the package.

4.2.5.3 Vendor Unique Name
This field contains a non-localised vendor name. It is used to check whether a package is a valid upgrade, during installation.

4.2.5.4 Names
This field contains an array of names of the SIS file. There must be exactly one name for each of the languages supported, and each name is matched to the corresponding language identified in the SISSupportedLanguages field of the SISController, at the same position in that array.

4.2.5.5 Vendor Names
This field contains an array of names of the SIS file vendor. There must be exactly one name for each of the languages supported, and each name is matched to the corresponding language identified in the SISSupportedLanguages field of the SISController, at the same position in that array.

4.2.5.6 Version
This field contains the version of the SIS file.

4.2.5.7 Creation Time
This field contains the creation time and date of the SIS file. However, this is not a secure timestamp and can easily be altered by the user changing their PC clock before creating the SIS file.

4.2.5.8 Install Type
This field contains the type of installation of the SIS file. Depending on the value, Software Install will install the package using different behaviours. The value is stored as a TUint8 but corresponds to the following enumeration:

```cpp
enum TInstallType {
    EInstInstallation,
    EInstAugmentation,
    EInstPartialUpgrade,
    EInstPreInstalledApp,
    EInstPreInstalledPatch
};
```

4.2.5.8.1 EInstApplication [SA]
The SIS file contains an application that can be installed on the device. Once it has been installed, it appears in the list of installed SIS files so that the user can remove it.

If the user wants to install a SIS Installation File that has the same UID and type EInstApplication on a device where there is already a SIS file installed with that UID and type EInstApplication then this is considered as an upgrade. The current version will be removed from the device and the new one will be installed.
4.2.5.8.2 EInstAugmentation [SP]
The SIS file contains an augmentation of an existing package. These files can be removed at a later
date, separately from the augmented application. This allows, for example, game levels which augment
an already installed application.

4.2.5.8.3 EInstPartialUpgrade [PU]
The SIS file contains a partial upgrade to an application. A partial upgrade to an application differs from a
normal application upgrade, in that the original package is not removed from the device, before the files
which are contained in the upgrade are installed. This allows, for example, a very small upgrade SIS to
replace just the parts of a package which require replacement, and not require re-delivery of the whole
package again.

4.2.5.8.4 EInstPreInstalledApp [PA]
This is a special indicator for use with applications which are pre-installed, in-place, on media cards.

4.2.5.8.5 EInstPreInstalledPatch [PP]
This is a special indicator for use with applications which are pre-installed, and augment another
application on the device (e.g. provide extra game levels).

4.2.5.9 Install Flags
This field contains flags which affect the installation. The value is stored as a TUint8 but corresponds to
the following enumeration:

```cpp
class TInstallFlags {
  EInstFlagShutdownApps = 1  // Shutdown all applications before uninstalling files
}
```

4.2.6 SISSupportedLanguages
This SISField contains an array of languages that the SIS file supports.

<table>
<thead>
<tr>
<th>Languages</th>
<th>SISArray&lt;SISSupportedLanguage&gt;</th>
</tr>
</thead>
</table>

4.2.7 SISSupportedOptions
This SISField contains options that the SIS file supports. The user is asked to select from these
options during install.

<table>
<thead>
<tr>
<th>Options</th>
<th>SISArray&lt;SISSupportedOption&gt;</th>
</tr>
</thead>
</table>

4.2.7.1 Options
This field is an array of options supported by the SIS file. There is one entry in the array for each option
supported by the SIS file, and its size may be zero or greater.
4.2.8 SISSupportedOption
This **SISField** contains names for a supported option of the SIS file. There is a name in the array for each supported language in the SIS file, in the same order as specified in the **SISSupportedLanguages SISField**.

<table>
<thead>
<tr>
<th>SISSupportedOption</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>SISArray&lt;SISString&gt;</td>
</tr>
</tbody>
</table>

4.2.9 SISPrerequisites
This **SISField** indicates the prerequisites that have to be met before Software Install will install the SIS file. The supported types of prerequisites are:

- SIS packages already installed on the device, and their version.
- The device must be one of a list of devices, identified by SIS files pre-installed on the device.

<table>
<thead>
<tr>
<th>SISPrerequisites</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Devices</td>
<td>SISArray &lt;SISDependency&gt;</td>
</tr>
<tr>
<td>Dependencies</td>
<td>SISArray &lt;SISDependency&gt;</td>
</tr>
</tbody>
</table>

4.2.9.1 Target Devices
This field is an array of **SISDependency SISFields** indicating on which devices this SIS file can be installed. Each device has a SIS file pre-installed, specific to that device. If the Target Devices **SISArray** contains any **SISDepencdencies** then at least one of these dependencies must be present in order to install the SIS file on the device. If the Target Devices **SISArray** has no entries then the SIS file can be installed on any type of device.

4.2.9.2 Dependencies
This field is an array of **SISDepencdencies** indicating which SIS packages need to be installed in order for this one to be installable. There maybe zero or more dependencies. For installation to continue, all the SIS files present in this **SISArray** must exist on the device.

4.2.10 SISSDependency
This **SISField** specifies a SIS package that must be installed on the device.

<table>
<thead>
<tr>
<th>SISSDependency</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>SISUid</td>
</tr>
<tr>
<td>Version Range</td>
<td>SISVersionRange</td>
</tr>
<tr>
<td>Dependency Names</td>
<td>SISArray&lt;SISString&gt;</td>
</tr>
</tbody>
</table>

4.2.10.1 UID
This field indicates the UID of the SIS package which needs to be installed on the device, in order to satisfy this dependency.
4.2.10.2 Version Range
This field indicates the range of versions of the SIS package that needs to be installed on the device. This SISField is optional; if it is not present then any version of the SIS package installed on the device will meet this dependency.

4.2.10.3 Dependency Names
This array contains the list of names of the dependency in each of the languages supported by the SIS file. There must be exactly one SISString per language supported by the SIS file.

4.2.11 SISProperties
The SISProperties block contains properties for the SIS package. These used to be called capabilities in the old format, but have been renamed to avoid confusion with platform security capabilities.

<table>
<thead>
<tr>
<th>SISProperties</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>SISArray&lt;SISProperty&gt;</td>
</tr>
</tbody>
</table>

4.2.12 SISProperty
This SISField contains a property, which is a key, value pair associated with a SIS package.

<table>
<thead>
<tr>
<th>SISProperty</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>TInt32 4 bytes</td>
</tr>
<tr>
<td>Value</td>
<td>TInt32 4 bytes</td>
</tr>
</tbody>
</table>

4.2.13 SISLogo
This SISField possibly contains a logo that is displayed during the installation progress.

<table>
<thead>
<tr>
<th>SISLogo</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo file</td>
<td>SISFileDescription</td>
</tr>
</tbody>
</table>

4.2.13.1 Logo file
This field contains the SISFileDescription for a logo file which is displayed at the start of installation. The MIME type field of the SISFileDescription is used to determine what format the logo is in. If the target field of the SISFileDescription is not an empty string, then the logo is also installed on the device.

4.2.14 SISFileDescription
This SISField gives information about a file stored in the SISData section.

<table>
<thead>
<tr>
<th>SISFileDescription</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>SISString</td>
</tr>
<tr>
<td>MIME Type</td>
<td>SISString</td>
</tr>
<tr>
<td>Capabilities</td>
<td>SISCapabilities</td>
</tr>
</tbody>
</table>
### 4.2.14.1 Target
This field is the location to install the file to. This is only used for the instructions that are actually going to copy the file somewhere on the device; it may be an empty string indicating the file will not be installed, for example when you want to run a file, or display it as a logo, without installing it on the device.

### 4.2.14.2 MIME Type
This field is the MIME type of the file described. This is used when running a file by MIME type and also when displaying an image file during install in order to choose the type of image decoder to use.

### 4.2.14.3 Capabilities
This is an optional SISField, and is only present if the file this SISFileDescription refers to is an executable file. The SISCapabilities SISField contains information about the capabilities the executable being described has been allocated.

### 4.2.14.4 Hash
This field contains the hash of the uncompressed file data.

### 4.2.14.5 Operation
This field is used to indicate how to process this file during installation.

```c
enum TsisFileOperation
{
    EOpInstall = 1, // Install File
    EOpRun = 2, // Run File
    EOpText = 4, // Display File as Text
    EOpNull = 8 // File is not present but will be removed on uninstall
};
```

### 4.2.14.6 Operation Options
This field indicates which options are applicable to the processing of this file during installation. The operation being carried out determines which options are valid.

#### 4.2.14.6.1 Options valid for EOpInstall

```c
enum TsisFileOperationOption
{
    EInstVerifyOnRestore = 1<<15 // Verify on Restore
};
```

`EInstVerifyOnRestore`
This option is used for secure backup and restore, to indicate that this file is not written-to after install, and so its contents should remain identical as to when it was installed. This allows verification, by checking the hash, upon restoration of the file from a backup.
4.2.14.6.2 Options valid for EOpRun

enum TInstFileRunOption
{
    EInstFileRunOptionInstall = 1<<1, // Run at installation
    EInstFileRunOptionUninstall = 1<<2, // Run at uninstallation
    EInstFileRunOptionByMimeType = 1<<3, // Run using MIME type
    EInstFileRunOptionWaitEnd = 1<<4, // Wait for end before continuing
    EInstFileRunOptionSendEnd = 1<<5, // Terminate after (un)install ends
};

EInstFileRunOptionInstall
This option indicates that the file specified will be run at installation time. If the target field is valid, then this file is installed to that location, otherwise this file is not copied to the device.

EInstFileRunOptionUninstall
This option indicates that the file specified will be run at uninstallation time. The target field must be valid, as Software Install will copy this file to the device so that it can be run at the time when the package is uninstalled.

EInstFileRunOptionByMimeType
This option indicates that the file is to be run, either at installation or uninstallation time, by MIME type. If this option is not set then the file specified will be run as an executable.

EInstFileRunOptionWaitEnd
If this option is set Software Install waits until the application being run finishes before continuing. Software Install should implement a sensible timeout however otherwise a malicious or malformed application could run forever and prevent any other access to Software Install without rebooting the device.

If this option is not set Software Install does not wait for the application being run to finish before continuing.

This option must not be set if EInstFileRunOptionSendEnd is set.

EInstFileRunOptionSendEnd
If this option is set then Software Install terminates this application, if it is still running, after the installation has finished.

This option must not be set if EInstFileRunOptionWaitEnd is set.

4.2.14.6.3 Options valid for EOpText

enum TInstTextOption
{
    EInstFileTextOptionContinue = 1<<9, // Continue button
    EInstFileTextOptionSkipIfNo = 1<<10, // Yes/No - skip next file if user selects no
    EInstFileTextOptionAbortIfNo = 1<<11, // Yes/No - abort install if user selects no
    EInstFileTextOptionExitIfNo = 1<<12, // Yes/No - uninstall if user selects no
};

EInstFileTextOptionContinue
This indicates that the installer should display the text, with a button to continue the install. After the dialog has been dismissed the installation will continue.
**EInstFileTextOptionSkipIfNo**

This indicates that the installer should display the text, with two buttons, one labelled yes and one labelled no. If the no button is pressed then the installer shall skip the file currently being processed, otherwise installation will continue as normal.

**EInstFileTextOptionAbortIfNo**

This indicates that the installer should display the text, with two buttons, one labelled yes and one labelled no. If the no button is pressed then the installer shall abort the installation, otherwise installation will continue as normal. The installer will display a dialog indicating that the installation has been aborted.

**EInstFileTextOptionExitIfNo**

This indicates that the installer should display the text, with two buttons, one labelled yes and one labelled no. If the no button is pressed then the installer shall abort the installation, otherwise installation will continue as normal. The only difference between this option and **EInstFileTextOptionAbortIfNo** is that the installer will not display a dialog indicating that the installation has been aborted.

### 4.2.14.7 Length

This field contains the length of the compressed file data the **SISFileDescription** is referring too in the SIS file itself.

### 4.2.14.8 Uncompressed Length

The length of the compressed file data the **SISFileDescription** is referring to, after it has been decompressed.

### 4.2.14.9 File Index

This is index of the **SISFileData SISField**, which contains the actual file data, in the Data Units field of the **SISDataUnit**.

### 4.2.15 SISCapabilities

This **SISField** represents the capabilities an executable within the SIS file has.

<table>
<thead>
<tr>
<th>SISHash</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capabilities</td>
<td>Variable length</td>
</tr>
</tbody>
</table>

#### 4.2.15.1 Capabilities

This is a bit-field containing all the capabilities the executable has been allocated. The field has variable length, but the length will always be a multiple of 4 bytes. It will be the only field in this **SISField**, and thus will have a length equal to the length of the **SISField**, given in the header. The lowest order bit will indicate whether the executable has been allocated capability 0 in the **TCapability** enumeration, the second lowest bit, capability 1, and so on.

### 4.2.16 SISHash

This **SISField** represents a hash.

<table>
<thead>
<tr>
<th>SISHash</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash Algorithm</td>
<td>TUInt32</td>
</tr>
<tr>
<td>Hash Data</td>
<td>SISBlob</td>
</tr>
</tbody>
</table>
4.2.16.1 Hash Algorithm
This field indicates the algorithm used to generate the hash. The following hash algorithms are currently supported:

```c
enum TSISHashAlgorithm {
    ESISHashAlgSHA1 = 1 // SHA-1 hash algorithm
}
```

4.2.16.2 Hash Data
This field contains the data of the hash contained in a `SI SBlob` `SI SField`. The length of this data depends upon the hashing algorithm used.

4.3 Signatures
The new SIS file format has been designed to support signing using multiple certificate chains. Multiple signatures are also supported for each chain, enabling different algorithms to be used for each of the signatures. Only one of these signatures needs to be validated for Software Install to consider the Certificate Chain as valid.

![Figure 3 – Diagram of signature and certificate chain layout in SIS file](image)

4.3.1 SISSignatureCertificateChain
This `SI SField` contains the signatures used to sign the SIS file and the certificate chain needed to validate the signatures.

<table>
<thead>
<tr>
<th>SISSignatureCertificateChain</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signatures</td>
<td>SISArray&lt;SISSignature&gt;</td>
</tr>
<tr>
<td>Certificate Chain</td>
<td>SISCertificateChain</td>
</tr>
</tbody>
</table>
4.3.1.1 Signatures
This field contains an array of signatures.

4.3.1.2 Certificate Chain
This field contains the certificate chain needed to verify the signatures.

4.3.2 SISCertificateChain
This SISField contains the certificate data as an ASN.1 encoded X509 certificate chain.

<table>
<thead>
<tr>
<th>SISCertificateChain</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Data</td>
<td>SISBlob</td>
</tr>
</tbody>
</table>

4.3.2.1 Certificate Data
This field contains the certificate data as an ASN.1 encoded X509 certificate chain.

4.3.3 SISSignature
This SISField contains the signature and an identifier of the signing and hashing algorithms used to generate it.

<table>
<thead>
<tr>
<th>SISSignature</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature Algorithm</td>
<td>SISSignatureAlgorithm</td>
</tr>
<tr>
<td>Signature Data</td>
<td>SISBlob</td>
</tr>
</tbody>
</table>

4.3.3.1 Signature Algorithm
This contains the algorithm used for signing, and the algorithm used for hashing the data, to enable the signature to be validated.

4.3.3.2 Signature Data
This field contains a SISBlob SISField containing the signature data.

4.3.4 SISSignatureAlgorithm
This SISField contains details about the signature and hash algorithms used to create a signature.

<table>
<thead>
<tr>
<th>SISSignatureAlgorithm</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm Identifier</td>
<td>SISString</td>
</tr>
</tbody>
</table>

4.3.4.1 Algorithm Identifier
This is a string delimited by '.' characters which represents the Object Identifier of the algorithms used. Currently supported algorithms are:

- "1.2.840.113549.1.1.5" - SHA-1 with RSA signature
- "1.2.840.10040.4.3" - SHA-1 with DSA signature
4.4 Expressions

The SIS file is generated from a textual package description. This description supports a simple format of deciding which files to install, at installation time, using if, then, and else constructs. This is encoded into the SIS package using the following SISFields.

4.4.1 SISIf

This SISField represents an if statement and condition in the package file used to generate the SIS file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>SISExpression</td>
</tr>
<tr>
<td>Install Block</td>
<td>SISInstallBlock</td>
</tr>
<tr>
<td>Else ifs</td>
<td>SISArray&lt;SISElseIf&gt;</td>
</tr>
</tbody>
</table>

4.4.1.1 Expression

This field contains the expression which is evaluated during the processing of this SISField during install.

4.4.1.2 Install Block

This field contains the SISInstallBlock that is processed recursively if the expression evaluates to true.

4.4.1.3 Else ifs

If the expression evaluates to false then each of these SISElseIf SISFields are evaluated in sequence. If one of the expressions evaluates to true then the SISInstallBlock belonging to the expression is processed recursively and no further SISElseIf blocks in the array are checked. There may be zero or greater SISElseIf SISFields in this array.

MakeSIS can simulate an else statement in the package, by adding a SISElseIf SISField, with a condition which always evaluates to true.

4.4.2 SISElseIf

This SISField represents the ‘else if’ part of an ‘if’ statement in the package file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>SISExpression</td>
</tr>
<tr>
<td>Install Block</td>
<td>SISInstallBlock</td>
</tr>
</tbody>
</table>

4.4.2.1 Expression

This SISExpression is evaluated by Software Install while processing the SISElseIf SISField.

4.4.2.2 Install Block

If Expression evaluates to true then this SISInstallBlock SISField is processed recursively by Software install.
4.4.3 SISInstallBlock

This SISField contains a list of files in the package which need to be installed, a list of embedded SIS files, and a list of SISIf blocks inside this install block. Each of these arrays may have zero or more entries.

<table>
<thead>
<tr>
<th>SISInstallBlock</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>SISArray&lt;SISFileDescription&gt;</td>
</tr>
<tr>
<td>Embedded SIS Files</td>
<td>SISArray&lt;SISController&gt;</td>
</tr>
<tr>
<td>If blocks</td>
<td>SISArray&lt;SISIf&gt;</td>
</tr>
</tbody>
</table>

4.4.3.1 Files

This field contains a list of files, which need to be processed with the SISInstallBlock. The most common operation to perform will be to install these files, but depending on the options they may be displayed to the user or run, see SISFileDescription for more information. There may be zero or greater SISFileDescription SISFields in this array.

4.4.3.2 Embedded SIS Files

This field contains a list of embedded SIS files, which are represented by SISControllers stored in the meta-data of the SIS file and need to be processed with the SISInstallBlock. There may be zero or greater SISController SISFields in this array.

4.4.3.3 If blocks

This field contains a list of SISIf fields, which need to be processed with the SISInstallBlock. Software Install will check the condition of each of these SISIf blocks and if it is true, process that SISIf block recursively. There may be zero or greater SISIf SISFields in this array.

4.4.4 SISExpression

This SISField represents an expression. Expressions are broken down into parts, and the whole expression is represented as a tree of SISExpression SISFields.

<table>
<thead>
<tr>
<th>SISExpression</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>TUint32</td>
</tr>
<tr>
<td>Integer Value</td>
<td>TInt32</td>
</tr>
<tr>
<td>String Value</td>
<td>SISString</td>
</tr>
<tr>
<td>Left Expression</td>
<td>SISExpression</td>
</tr>
<tr>
<td>Right Expression</td>
<td>SISExpression</td>
</tr>
</tbody>
</table>

4.4.4.1 Operator

The Operator field indicates what the operator is for this expression and thus determines which of the other fields are valid.

```cpp
enum TOperator
{
    // Binary Operators
    EBinOpEqual = 1,    // equal to
    EBinOpNotEqual,    // not equal to
    EBinOpGreaterThan, // greater than
```
EBinOpLessThan, // less than
EBinOpGreaterOrEqual, // greater than or equal to
EBinOpLessOrEqual, // less than or equal to

// Logical Operators
ELogOpAnd, // logical AND
ELogOpOr, // logical OR

// Unary Operators
EUnaryOpNot, // NOT() - logical NOT

// Functions
EFuncExists, // EXISTS() - Checks if the file exists
EFuncAppProperties, // APPPROP() - Queries application properties
EFuncDevProperties, // PACKAGE() - Queries for an installed package

// Primitives
EPrimTypeString, // This expression holds a string value
EPrimTypeOption, // This expression is an option, identified by integer
EPrimTypeVariable, // This expression is a variable, identified by integer
EPrimTypeNumber // This expression holds a number value

4.4.4.1.1 Function Descriptions

EXISTS(string1) (EFuncExists)
This function takes a string value, indicating a file name. During install if the file is present on the device then this function returns ETrue, otherwise it returns EFalse.

If the operator is EFuncExists, then this SISExpression will contain a SISString SISField containing the name of the file to check for at installation time.

APPPROP(expression1, expression2) (EFuncAppProperties)
This function queries properties for an installed SIS file. A property is a key, value pair. The first parameter is the UID of the installed SIS file, or the SIS file currently being installed. The second parameter is the key of the key, value pair. This function returns the value integer corresponding to the key or 0 if the key was not found, or if either of the expressions does not evaluate to an integer value. The keys must be unique in a particular SIS file so multiple values will never be found.

If the operator is EFuncAppProperties, then this SISExpression will contain Left, and Right SISExpressions. The Left field corresponds to the first parameter and the Right field gives the second parameter.

PACKAGE(expression1) (EFuncDevProperties)
This function queries for the existence of a package installed on the device. The integer parameter given is the pUID to search for. It returns ‘1’ if the package was found, and zero otherwise.

If the operator is EFuncDevProperties, then this the Left Expression will be present containing SISExpression.

4.4.4.1.2 Variables
The SIS file format supports the use of variables when creating expressions. Most of these variables correspond to device properties, which can be queried by using the HAL::Get() function on the device. The other variable which is supported is ‘Language’, which takes on the value of the language selected
by the user, at install time, and 'Remotelnstall'. For more information about these variables, see Appendix B. Variables are stored using an integer corresponding to the following enumeration:

```c
enum TVariableIndex
{
    // 0–0x1000 are reserved for HALData values, see Appendix B for the allowed values
    EVarLanguage = 0x1001,
    EVarRemotelnstall = 0x1002
}
```

### 4.4.4.2 Integer Value

This part of the expression can contain an integer value. It will be valid if the type of the expression is `EPrimTypeNumber`, `EPrimTypeVariable`, or `EPrimTypeOption`.

### 4.4.4.3 String Value

This part of the expression can contain a string. This field is optional; it will be present only if the type of the expression is `EPrimTypeString`, or `EFuncExists`.

### 4.4.4.4 Left Expression

This is the left sub-part of the expression. This field is optional; it will be present when the operator of this `SISExpression` is any binary operators `EBinOpEqual`, `EBinOpNotEqual`, `EBinOpGreaterThan`, `EBinOpLessThan`, `EBinOpLessOrEqual`, `EBinOpGreaterOrEqual`, or the logical operators `ELogOpAnd` and `ELogOpOr`, or the function operators `EFuncAppProperties` and `EFuncDevProperties`.

### 4.4.4.5 Right Expression

This is the right sub-part of the expression. This field is optional; it will be present when the operator of this `SISExpression` is not any of the primitives `EPrimTypeString`, `EPrimTypeOption`, `EPrimTypeVariable`, or `EPrimTypeNumber`, or the function `EFuncExists`.

### 4.5 SIS File Data

This section of the SIS file contains the actual file data that is used during the install process. It consists of an array of data units, each of which contains the files from one `SISController`. There may be more than one data unit if there are embedded SIS files. Each `SISController` has a field containing the index into the Data Units array in the SISData `SISField`. This contains the files which are installed by that `SISController`. This makes it easy to add and remove embedded SIS files.
4.5.1 SISData

The SISData SISField contains all of the file data for a SIS file.

<table>
<thead>
<tr>
<th>SISData</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Units</td>
<td>SISArray&lt;SISDataUnit&gt;</td>
</tr>
</tbody>
</table>

4.5.1.1 Data Units

There is one data unit present for each SISController in the meta-data of the SIS file. There may be more than one SISController, and thus data unit, if there are embedded SIS files.

4.5.2 SISDataUnit

The SISDataUnit contains all the file data for a SISController.

<table>
<thead>
<tr>
<th>SISDataUnit</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Data</td>
<td>SISArray&lt;SISFileData&gt;</td>
</tr>
</tbody>
</table>
4.5.2.1 File Data
This field is an array of possibly compressed SISFileData SISFields. There is an entry in this array for every file which it is possible for this SISController to install.

4.5.3 SISFileData
The SISFileData SISField contains the actual data for a file, which may be compressed.

<table>
<thead>
<tr>
<th>SISFileData</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Data</td>
<td>SISCompressed&lt;Raw File Data&gt;</td>
</tr>
</tbody>
</table>

4.5.3.1 File Data
This field contains the possibly compressed file data. The raw file data is stored in the Compressed Data field of the SISCompressed SISField.
### Appendix A - ESISField Type Values

<table>
<thead>
<tr>
<th>ESISField</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid ESISField</td>
<td>0</td>
</tr>
<tr>
<td>SISString</td>
<td>1</td>
</tr>
<tr>
<td>SISArray</td>
<td>2</td>
</tr>
<tr>
<td>SISCompressed</td>
<td>3</td>
</tr>
<tr>
<td>SISVersion</td>
<td>4</td>
</tr>
<tr>
<td>SISVersionRange</td>
<td>5</td>
</tr>
<tr>
<td>SISDate</td>
<td>6</td>
</tr>
<tr>
<td>SISTime</td>
<td>7</td>
</tr>
<tr>
<td>SISDateTime</td>
<td>8</td>
</tr>
<tr>
<td>SISUid</td>
<td>9</td>
</tr>
<tr>
<td>Unused</td>
<td>10</td>
</tr>
<tr>
<td>SISLanguage</td>
<td>11</td>
</tr>
<tr>
<td>SISContents</td>
<td>12</td>
</tr>
<tr>
<td>SISController</td>
<td>13</td>
</tr>
<tr>
<td>SISInfo</td>
<td>14</td>
</tr>
<tr>
<td>SISSupportedLanguages</td>
<td>15</td>
</tr>
<tr>
<td>SISSupportedOptions</td>
<td>16</td>
</tr>
<tr>
<td>SISPrerequisites</td>
<td>17</td>
</tr>
<tr>
<td>SISDependency</td>
<td>18</td>
</tr>
<tr>
<td>SISProperties</td>
<td>19</td>
</tr>
<tr>
<td>SISProperty</td>
<td>20</td>
</tr>
<tr>
<td>SISignatures</td>
<td>21</td>
</tr>
<tr>
<td>SISCertificateChain</td>
<td>22</td>
</tr>
<tr>
<td>SISLogo</td>
<td>23</td>
</tr>
<tr>
<td>SISFileDescription</td>
<td>24</td>
</tr>
<tr>
<td>SISHash</td>
<td>25</td>
</tr>
<tr>
<td>SISlf</td>
<td>26</td>
</tr>
<tr>
<td>SISElseIf</td>
<td>27</td>
</tr>
<tr>
<td>SISInstallBlock</td>
<td>28</td>
</tr>
<tr>
<td>SISExpression</td>
<td>29</td>
</tr>
<tr>
<td>SISData</td>
<td>30</td>
</tr>
<tr>
<td>SISDataUnit</td>
<td>31</td>
</tr>
<tr>
<td>SISFileData</td>
<td>32</td>
</tr>
<tr>
<td>SISSupportedOption</td>
<td>33</td>
</tr>
<tr>
<td>SISControllerChecksum</td>
<td>34</td>
</tr>
<tr>
<td>SISDataChecksum</td>
<td>35</td>
</tr>
<tr>
<td>SISSignature</td>
<td>36</td>
</tr>
<tr>
<td>SISBlob</td>
<td>37</td>
</tr>
<tr>
<td>SISSignatureAlgorithm</td>
<td>38</td>
</tr>
<tr>
<td>SISSignatureCertificateChain</td>
<td>39</td>
</tr>
<tr>
<td>SISDataCertificateChain</td>
<td>40</td>
</tr>
<tr>
<td>SISCapabilities</td>
<td>41</td>
</tr>
</tbody>
</table>
## Appendix B - Variable Names

The SIS and package file formats allow complex expressions to be built up, which determine at install time, whether files are copied to the device. Software Install will assign values to these variables at installation time.

The following variables are provided, which can be used to get information about the device. Each of these corresponds to the value returned by `HALData::Get()` with different attributes.

<table>
<thead>
<tr>
<th>Variable Name (Keyword specified in .pkg File)</th>
<th>Attribute used by HALData::Get (In C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>HALData::EManufacturer</td>
</tr>
<tr>
<td>ManufacturerHardwareRev</td>
<td>HALData::EManufacturerHardwareRev</td>
</tr>
<tr>
<td>ManufacturerSoftwareRev</td>
<td>HALData::EManufacturerSoftwareRev</td>
</tr>
<tr>
<td>ManufacturerSoftwareBuild</td>
<td>HALData::EManufacturerSoftwareBuild</td>
</tr>
<tr>
<td>Model</td>
<td>HALData::EModel</td>
</tr>
<tr>
<td>MachineUid</td>
<td>HALData::EMachineUid</td>
</tr>
<tr>
<td>DeviceFamily</td>
<td>HALData::EDeviceFamily</td>
</tr>
<tr>
<td>DeviceFamilyRev</td>
<td>HALData::EDeviceFamilyRev</td>
</tr>
<tr>
<td>CPU</td>
<td>HALData::ECPU</td>
</tr>
<tr>
<td>CPUArch</td>
<td>HALData::ECPUArch</td>
</tr>
<tr>
<td>CPUABI</td>
<td>HALData::ECPUABI</td>
</tr>
<tr>
<td>CPUSpeed</td>
<td>HALData::ECPUSpeed</td>
</tr>
<tr>
<td>SystemTickPeriod</td>
<td>HALData::ESystemTickPeriod</td>
</tr>
<tr>
<td>MemoryRAM</td>
<td>HALData::EMemoryRAM</td>
</tr>
<tr>
<td>MemoryRAMFree</td>
<td>HALData::EMemoryRAMFree</td>
</tr>
<tr>
<td>MemoryROM</td>
<td>HALData::EMemoryROM</td>
</tr>
<tr>
<td>MemoryPageSize</td>
<td>HALData::EMemoryPageSize</td>
</tr>
<tr>
<td>PowerBackup</td>
<td>HALData::EPowerBackup</td>
</tr>
<tr>
<td>Keyboard</td>
<td>HALData::EKeyboard</td>
</tr>
<tr>
<td>KeyboardDeviceKeys</td>
<td>HALData::EKeyboardDeviceKeys</td>
</tr>
<tr>
<td>KeyboardAppKeys</td>
<td>HALData::EKeyboardAppKeys</td>
</tr>
<tr>
<td>KeyboardClick</td>
<td>HALData::EKeyboardClick</td>
</tr>
<tr>
<td>KeyboardClickVolumeMax</td>
<td>HALData::EKeyboardClickVolumeMax</td>
</tr>
<tr>
<td>DisplayXPixels</td>
<td>HALData::EDisplayXPixels</td>
</tr>
<tr>
<td>DisplayYPixels</td>
<td>HALData::EDisplayYPixels</td>
</tr>
<tr>
<td>DisplayXTwips</td>
<td>HALData::EDisplayXTwips</td>
</tr>
<tr>
<td>DisplayYTwips</td>
<td>HALData::EDisplayYTwips</td>
</tr>
<tr>
<td>DisplayColors</td>
<td>HALData::EDisplayColors</td>
</tr>
<tr>
<td>DisplayContrastMax</td>
<td>HALData::EDisplayContrastMax</td>
</tr>
</tbody>
</table>
The following additional variables provide information about the choices of the user during installation.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>This variable will be set to the language the user chooses. The integer value will correspond to the TLanguage enumeration.</td>
</tr>
<tr>
<td>optionN (where N is a number from 1 to the number of options in the SIS file).</td>
<td>There is a variable for each of the different options in the SIS file. The value of each of these variables will be 1 if the option was selected and 0 otherwise.</td>
</tr>
</tbody>
</table>

Back to Developer Library

Want to be kept informed of new articles being made available on the Symbian Developer Network? Subscribe to the Symbian Community Newsletter.

The Symbian Community Newsletter brings you, every month, the latest news and resources for Symbian OS.