Annex A15: Hierarchy Information
**Table 0  Revision History**

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>5/17/2009</td>
<td>Various comments incorporated</td>
</tr>
<tr>
<td>1.1</td>
<td>4/28/2009</td>
<td>Add LocalChangeBit section and Template File compliance statement</td>
</tr>
<tr>
<td>1.0</td>
<td>3/22/2009</td>
<td>Draft for publication</td>
</tr>
</tbody>
</table>

**LEGAL DISCLAIMER**

“This specification provided “AS IS” and without any warranty of any kind, including, without limitation, any express or implied warranty of non-infringement, merchantability or fitness for a particular purpose.

In no event shall IBTA or any member of IBTA be liable for any direct, indirect, special, exemplary, punitive, or consequential damages, including, without limitation, lost profits, even if advised of the possibility of such damages.”
# Table of Contents

Annex A15: Hierarchy Information ................................................................. 8
A15.1 Introduction ......................................................................................... 8
A15.2 Glossary ............................................................................................. 8
A15.3 Overview ............................................................................................ 8
A15.4 Mode of Operation ............................................................................ 12
  A15.4.1 Storing and Updating Hierarchical Information ....................... 12
  A15.4.2 Fabric Discovery .......................................................................... 12
  A15.4.3 Decoding Hierarchical Information ............................................ 13
A15.5 Subnet Management Extensions ....................................................... 13
  A15.5.1 Hierarchy Information Capability ............................................. 14
  A15.5.2 Hierarchy Information OtherLocalChanges bit ....................... 14
  A15.5.3 HierarchyInfo Attribute .............................................................. 15
  A15.5.4 Error handling for HierarchyInfo Attribute .............................. 17
A15.6 Subnet Administration Extensions .................................................... 19
  A15.6.1 HierarchyInfo Support ................................................................. 19
  A15.6.2 HierarchyInfoRecord Attribute ................................................ 19
A15.7 Hierarchy Template File ................................................................. 20
  A15.7.1 Key Schema Design Considerations ....................................... 20
  A15.7.2 Schema Elements ....................................................................... 20
  A15.7.3 RecordSelector, RecordCode and ApplyToLevels Elements ........ 23
  A15.7.4 Formal Template File XML Schema .......................................... 23
  A15.7.5 A Generic Physical Hierarchy Template .................................... 26
  A15.7.6 A Generic Port Hierarchy Template .......................................... 28
A15.8 Compliance ....................................................................................... 31
<p>| Figure 1 | Equipment Rack ................................................................. | 9 |
| Figure 2 | Flat topology view.............................................................. | 10 |
| Figure 3 | Physical hierarchy view ..................................................... | 10 |
| Figure 4 | Hierarchy Template File XML Schema .................................... | 24 |
| Figure 5 | Generic Physical Hierarchy Template ................................... | 26 |
| Figure 6 | Physical hierarchy view ..................................................... | 27 |
| Figure 7 | Generic Port Hierarchy Template ........................................ | 29 |</p>
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Revision History</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>HierarchyInfo Attribute</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>HierarchyInfo Attribute / Method Map Errors</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>HierarchyInfo AttributeModifier Errors</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>HierarchyInfo Component Errors</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>HierarchyInfoRecord</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Physical HierarchyInfo GetResponse Example</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>Port HierarchyInfo GetResponse Example</td>
<td>30</td>
</tr>
</tbody>
</table>
ANNEX A15: HIERARCHY INFORMATION

A15.1 INTRODUCTION

This document is a supplement to Release 1.2.1 of Volume 1 of the InfiniBand Architecture, herein referred to as the base document. This document provides an annex to the base document that defines the framework for discovering the hierarchy of an InfiniBand fabric.

Support for this Hierarchy Information framework is optional, but if it is supported it must be done in accordance with the requirements of this annex.

A15.2 GLOSSARY

HI

See Hierarchy Information

Hierarchy Information

Information which progressively describes the fabric hardware components (Ports, CAs, switches, and routers) in a hierarchical manner.

HI Template

An XML file holding the encoding and semantics of the HI data returned from a node.

HI Template GUID

A unique identifier assigned to each HI Template.

Physical Hierarchy

The hierarchy representing the location of the InfiniBand node. Going from bottom up it will include the node location in the board, the slot that board is located in the chassis, the chassis location within the rack, the rack location within the room, etc.

Port Hierarchy

Describes the mapping between an InfiniBand device port to the front panel port or plug it connects to. Applying this mapping iteratively, the front panel port or plug can be further mapped to the port or plug of a higher level enclosure the system is embedded in.

Raw HI

Information as stored within the fabric nodes. Requires decoding by the corresponding HI Template to have meaning.

A15.3 OVERVIEW

InfiniBand Subnet Management infrastructure enables a Subnet Manager entity to discover the network connectivity. The discovered topology which is a graph of InfiniBand nodes (CA/Router/Switch) connected by InfiniBand links, can be used for reporting network issues and characteristics. Unlike many other technologies, the InfiniBand connectivity graph is com-
pletely discovered even if parts of it are enclosed within embedded systems.

For example, consider the equipment rack depicted in Figure 1 on page 9. The rack is built of 16 1U servers, at locations U1 to U16, and a sixteen-port InfiniBand switch system, at location U17.

The connectivity graph as would be discovered by the SM in its discovery phase is provided in Figure 2 on page 10. The 16 port InfiniBand switch is represented as a connectivity graph made of 6 switches of 8 ports each. We refer to this connectivity view as a 'flattened' view, since the system-level hierarchy is missing from it.
This strength of the InfiniBand standardization, i.e., of providing a common view of the network, has a drawback which the Hierarchy Information extension is intended to solve: humans work with physical systems, not InfiniBand abstractions.

As an example, assume the link highlighted in red in Figure 2 on page 10 located at directed path 1,3 from node H1 (SM node) has reported bad link stability. Reporting the link instability using the directed path of 1,3 requires some indirect translation to actually point to the link in trouble. However, if the report included the physical hierarchy information shown in Figure 3 on page 10, i.e., that this link is between InfiniBand switch SW1 of SLOT2 and InfiniBand SW2 of SLOT1 in switch chassis U17 which is located in rack 3, the fabric administrator knows immediately which physical boards and chassis are having a problem. Furthermore, the location of that chassis and boards, is directly provided.
The description above exemplifies a chassis or physical containment hierarchy. It is the objective of this annex to support not only that type, but to enable other, arbitrary hierarchy types as well. The Hierarchy Information Annex describes extensions to Infiniband Subnet Management and Subnet Administration which allow administrative tools to understand and display physical properties of systems in a flexible, user-focused manner.

The crux of the extension is standardizing how InfiniBand nodes report hierarchy information. There are two methods for the hierarchical information to be programmed into the node:

- By the SM, using SubnSet(HierarchyInfo)
- Using vendor specific embedded software which is beyond the scope of this specification

To support the concept of arbitrary hierarchy semantics, a new Hierarchy-Info attribute is used to manage the hierarchy information stored on the node, which must be decoded by applying a “Hierarchy Template”. The template provides semantic structure to the raw, unstructured data provided directly by the InfiniBand nodes.

The structure of the annex is as follows: in the next section we describe the flow of Hierarchy Information and the involved entities. The following two sections describe the enhancements to Subnet Management and Administration. The definition of the Hierarchy Template file including two common hierarchies (i.e., Physical-Hierarchy and Port-Hierarchy) as examples. The last section is the compliance statement summary.

To summarize, the annex goals are:

- To define a flexible, generic encoding of hierarchical information associated with ports and nodes in an InfiniBand network. This information can relate to physical location, grouping of devices within enclosures, connectivity, power, or any other information a vendor chooses to provide.
- To provide an encoding that is as space efficient as possible, since storage on actual devices is expensive. The device stores data in an encoded format which can be very space efficient, and the decoding of the data is provided by an external XML description of the data.
- To provide sample encoding of the data that real-world experience has shown to be vital to efficient network administration.
A15.4 MODE OF OPERATION

This section describes the flow of hierarchy-related information and the hardware and software involved. The overall process has the following steps which are described in details in the following sub sections:

- Storing and updating hierarchical information in InfiniBand nodes
- Fabric discovery and query of the raw hierarchical information
- Decoding hierarchical information using corresponding templates

A15.4.1 STORING AND UPDATING HIERARCHICAL INFORMATION

It is anticipated that a node is pre-programmed with its hierarchical information when it comes on-line and starts responding to subnet management packets. A vendor-dependent mechanism is assumed to provide that initial information. For the Physical Hierarchy, vendor-dependent embedded software will program each device in the chassis with information identifying its location therein.

Systems supporting hot-plug installation of subsystems are expected to include a mechanism for updating the hierarchy information contained in the new hardware.

Some of the hierarchical information to be associated with a device may not be known until it is in its final location in the system. In our Physical Hierarchy example Figure 3 on page 10 the location of the device chassis within the rack, and the rack location in the datacenter are examples of additional physical information that aid an administrator in locating and debugging network issues. The Hierarchy Information mechanisms allow vendors and administrators to supplement the hierarchy information provided by the embedded system either directly at the embedded system (via vendor-dependant mechanisms) or in-band via the SubnSet(HierarchyInfo) mechanism.

A15.4.2 FABRIC DISCOVERY

A Subnet Manager supporting the optional functionality described in this annex will recognize nodes supporting Hierarchy Information by examining the capability mask bits they return in response to SubnGet(Port-Info). For those nodes, subsequent calls to SubnGet(HierarchyInfo) will return their relevant hierarchy information.

In order to support multiple hierarchies, the SubnGet(HierarchyInfo) attribute modifier includes a hierarchy index which identifies the type of hierarchy queried. To enable storing per port information the attribute modifier used in query of switch devices also contains the target port number.
The capability mask bit and HierarchyInfo attribute defined by this annex are described in detail in Section A15.5, "Subnet Management Extensions," on page 13.

Discovered Hierarchical Information is made available by SA to clients with sufficient permissions. The corresponding HierarchyInfoRecord attribute and its usage is described in Section A15.6, "Subnet Administration Extensions," on page 19.

A15.4.3 DECODING HIERARCHICAL INFORMATION

Raw Hierarchical Information returned in response to a SubnGet(HierarchyInfo) query has two parts:

• A Hierarchy Information Template GUID
• Template-specific encoded data

The Hierarchy Information Template GUID is the key by which the matching Hierarchy Information Template File is found.

To support arbitrary hierarchies Hierarchy Template files take the form of XML, which defines the semantics and data encoding.

Two example templates are provided in this annex, one for generic physical hierarchy representation, the other for device to system port mapping.

The definition of the syntax for the Hierarchy Template file is provided in Section A15.7, "Hierarchy Template File," on page 20

A15.5 SUBNET MANAGEMENT EXTENSIONS

The following sections describe the Subnet Management class extensions for Hierarchy Information.

These extensions are:

• A new PortInfo:CapabilityMask bit declaring the SMA's support for HierarchyInfo attribute
• A new OtherLocalChanges bit in trap 144 data details to flag a change in a node hierarchy info
• HierarchyInfo attribute which returns the encoded raw hierarchy data
• Error conditions for the above attribute
A15.5.1 HIERARCHY INFORMATION CAPABILITY

The PortInfo:CapabilityMask component (described in Table 146, "Port-Info" on page 830) is extended by this annex. The previously reserved bit 31 of the PortInfo:CapabilityMask is now defined as IsHierarchyInfoSupported, declaring support for the HierarchyInfo attribute, specified in this annex.

HI15-1: A switch that sets PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one on its management port shall support the HierarchyInfo attribute on all its external ports.

A15.5.2 HIERARCHY INFORMATION OTHERLOCALCHANGES BIT

Trap 144, reporting local changes (described in Section 14.3.13, “Other Local Changes,” on page 870) is extended by this annex to report hierarchy information changes. A new bit named HierarchyInfoChange at bit 12 of the third reserved field (used for other local changes) of the Notice:DataDetails is defined and should be set to one to mark the notice as reporting hierarchy information changes.

The changes to Table 136, “Notice DataDetails For Trap 144” on page 823 are:

• The width of the third reserved field (used for other local changes), is decreased by 1 to 11
• A single bit field named HierarchyInfoChange is added to the table just below the third reserved field and above SMPriorityChange field.

HI15-2: If the port supports Traps as indicated in the PortInfo:CapabilityMask.IsTrapSupported, and the port is capable of sending the enhanced (for other local changes) version of Trap 144 as indicated in the PortInfo:CapabilityMask.IsOtherLocalChangesNoticeSupported, and the port supports Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, then the SMA shall send a Trap 144 to the SM when the hierarchy information is modified by an entity on the sending port’s endnode at runtime.

HI15-3: If the management port supports Notices as indicated in the PortInfo:CapabilityMask.IsNoticeSupported, and the port is capable of logging the OtherLocalChanges notice as indicated in the PortInfo:CapabilityMask.IsOtherLocalChangesNoticeSupported, and the port supports Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one the SMA shall log a Notice using Notice:TrapNumber 144 when the hierarchy information is modified by an entity on the sending port’s endnode at runtime.
A new Subnet Management attribute, HierarchyInfo (with Attribute ID 0x001E), is defined in this section. The HierarchyInfo Attribute provides port-specific hierarchy information.

**HI15-4**: An endpoint supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, shall respond to SubnGet(HierarchyInfo) and SubnSet(HierarchyInfo) according to the behavior described in the following section.

- The other bits 31:16 of the AttributeModifier are reserved.
- The AttributeModifier bits 15:8 define HierarchyIndex. This field is used to select a specific hierarchy maintained for the specified port. Each hierarchy supported by the port is identified with a HierarchyIndex value in the range zero through HierarchyInfo:MaxActiveIndex, as returned in the response to SubnGet(). The HierarchyIndex zero is always valid even if its data fields are unused, and thus can be queried to obtain the value of MaxActiveIndex and MaxSupportedIndex.
- The AttributeModifier bits 7:0 select the port upon which the operation specified by the SMP is performed. For switches, the range of values is between 0 to N, where N is the number of switch ports. For CAs and routers, these bits are reserved and the operation is performed on the port that receives the SMP.

Note that each switch external port may support a different set of hierarchies.

Note that there need be no relationship between the values of HierarchyIndex and TemplateGUID. The HierarchyIndex value does not imply anything about the meaning of the obtained hierarchy information and should only be used as means to retrieve the active hierarchies available on the port. No relation between the index and obtained TemplateGUID is guaranteed.

The Level0Data through Level12Data components of the HierarchyInfo attribute provide the encoded hierarchy data. Level0Data is the data for the lowest hierarchy level. Level1Data provides the data one level above it, and so on.
Table 1  HierarchyInfo Attribute

<table>
<thead>
<tr>
<th>Component</th>
<th>Access</th>
<th>Length (bits)</th>
<th>Offset (bits)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TemplateGUID</td>
<td>RW</td>
<td>64</td>
<td>0</td>
<td>A unique, vendor assigned identifier, that recognizes the Hierarchy Information Template to be used for decoding the hierarchy information provided by this attribute.</td>
</tr>
<tr>
<td>MaxSupportedIndex</td>
<td>RO</td>
<td>8</td>
<td>64</td>
<td>The largest possible hierarchy index number supported on the given port.</td>
</tr>
<tr>
<td>MaxActiveIndex</td>
<td>RO</td>
<td>8</td>
<td>72</td>
<td>Returns the largest valid hierarchy index value in use on the port. The SMA maintains this value in accordance with the number of active hierarchies.</td>
</tr>
<tr>
<td>ActiveLevels</td>
<td>RW</td>
<td>4</td>
<td>80</td>
<td>Total number of hierarchy levels provided in this attribute data. For SubnGet(), a value of zero notes the specific queried hierarchy was not setup on the port, thus it is unused. For SubnSet(), a value of zero invalidates the specific hierarchy information of the port.</td>
</tr>
<tr>
<td>Reserved</td>
<td>RO</td>
<td>4</td>
<td>84</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>RO</td>
<td>8</td>
<td>88</td>
<td>Reserved</td>
</tr>
<tr>
<td>Level0Data</td>
<td>RW</td>
<td>32</td>
<td>96</td>
<td>Level 0 encoded data of the specific port</td>
</tr>
<tr>
<td>Level1Data</td>
<td>RW</td>
<td>32</td>
<td>128</td>
<td>Level 1 encoded data of the specific port</td>
</tr>
<tr>
<td>Level2Data</td>
<td>RW</td>
<td>32</td>
<td>160</td>
<td>Level 2 encoded data of the specific port</td>
</tr>
<tr>
<td>Level3Data</td>
<td>RW</td>
<td>32</td>
<td>192</td>
<td>Level 3 encoded data of the specific port</td>
</tr>
<tr>
<td>Level4Data</td>
<td>RW</td>
<td>32</td>
<td>224</td>
<td>Level 4 encoded data of the specific port</td>
</tr>
<tr>
<td>Level5Data</td>
<td>RW</td>
<td>32</td>
<td>256</td>
<td>Level 5 encoded data of the specific port</td>
</tr>
<tr>
<td>Level6Data</td>
<td>RW</td>
<td>32</td>
<td>288</td>
<td>Level 6 encoded data of the specific port</td>
</tr>
<tr>
<td>Level7Data</td>
<td>RW</td>
<td>32</td>
<td>320</td>
<td>Level 7 encoded data of the specific port</td>
</tr>
<tr>
<td>Level8Data</td>
<td>RW</td>
<td>32</td>
<td>356</td>
<td>Level 8 encoded data of the specific port</td>
</tr>
<tr>
<td>Level9Data</td>
<td>RW</td>
<td>32</td>
<td>384</td>
<td>Level 9 encoded data of the specific port</td>
</tr>
<tr>
<td>Level10Data</td>
<td>RW</td>
<td>32</td>
<td>416</td>
<td>Level 10 encoded data of the specific port</td>
</tr>
<tr>
<td>Level11Data</td>
<td>RW</td>
<td>32</td>
<td>448</td>
<td>Level 11 encoded data of the specific port</td>
</tr>
<tr>
<td>Level12Data</td>
<td>RW</td>
<td>32</td>
<td>480</td>
<td>Level 12 encoded data of the specific port</td>
</tr>
</tbody>
</table>
A15.5.3.1 **HierarchyInfo Query**

The SM performing SubnGet(HierarchyInfo) is expected to first obtain the range of available hierarchy information sets on the given port by using zero for the value of HierarchyIndex, then to repeat the query using values from one up to HierarchyInfo:MaxActiveIndex obtained from the first query.

The value N of the resulting HierarchyInfo:ActiveLevels qualifies the first N (zero through N-1) Level<i>Data fields of the HierarchyInfo. If N equals zero it means the HierarchyInfo is invalid and should be ignored.

A15.5.3.2 **Setting HierarchyInfo**

A SM undertaking to update the hierarchy information in a node should first determine the number of hierarchies the node can support by querying it with SubnGet(HierarchyInfo), using zero as the value of HierarchyIndex. If the response to this query yields a HierarchyInfo:MaxSupportedIndex value of M, say, a performance optimization for the SM is to use values contiguously from zero through M in SubnSet(HierarchyInfo) to set up multiple hierarchy information data sets.

A HierarchyInfo attribute is valid only if its HierarchyInfo:ActiveLevels is greater than zero. To invalidate a particular HierarchyInfo attribute of a node a SM can perform SubnSet(HierarchyInfo) on that node with HierarchyInfo:ActiveLevels value of zero.

**HI15-5:** An endport supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, shall track for every port the maximal HierarchyIndex of valid HierarchyInfo attributes and report it in the subsequent SubnGet(HierarchyInfo) MaxActiveIndex component.

A15.5.4 **Error Handling for HierarchyInfo Attribute**

The following tables describe the error conditions for HierarchyInfo attribute.

A15.5.4.1 **HierarchyInfo Method/Attribute Combination Not Supported (status_field[4:2] = 0x3)**

The following table extends Table 165, “Subnet Management Attribute / Method Map Errors” on page 853.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Get</th>
<th>Set</th>
<th>Trap</th>
<th>TrapRepress</th>
</tr>
</thead>
<tbody>
<tr>
<td>HierarchyInfo</td>
<td>Valid</td>
<td>Valid</td>
<td>error</td>
<td>error</td>
</tr>
</tbody>
</table>
A15.5.4.2 **HierarchyInfo AttributeModifier Errors (status_field[4:2] = 0x7)**

The following table extends [Table 166, "SMP AttributeModifier Errors" on page 855](#).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invalid AttributeModifier for Switch with BSP0</th>
<th>Invalid AttributeModifier for CA/Router/Switch with ESP0</th>
</tr>
</thead>
</table>

---

A15.5.4.3 **HierarchyInfo Attribute Component Errors (status_field[4:2] = 0x7)**

The following table describes error conditions when inspecting specific components of the HierarchyInfo attribute.

<table>
<thead>
<tr>
<th>Component</th>
<th>Invalid Set() component value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveLevels</td>
<td>&gt; 13</td>
</tr>
</tbody>
</table>

---
A15.6 SUBNET ADMINISTRATION EXTENSIONS

This section describes the Subnet Administration class extension for Hierarchy Information including a new attribute, HierarchyInfoRecord.

A15.6.1 HIERARCHYINFO SUPPORT

A new SA capability mask bit is defined for ClassPortInfo:CapabilityMask2 extending Table 188, “SA-Specific Optional Capabilities” on page 899:

- Name: IsHierarchyInfoSupported
- Bit: 4
- Description: If this value is 1, SA shall support the HierarchyInfoRecord attribute as defined in Section A15.6.2, “HierarchyInfoRecord Attribute,” on page 19.

A15.6.2 HIERARCHYINFORECORD ATTRIBUTE

A new Subnet Administration attribute, HierarchyInfoRecord (with Attribute ID 0x003C), is defined in this section. This attribute encapsulates HierarchyInfo attribute and provides a means for SA clients to obtain the hierarchy information discovered by the SM.

HI15-6: If SA supports Hierarchy Information as indicated by setting its ClassPortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, it shall respond to SubnAdmGet() and SubnAdmGetTable() of the HierarchyInfoRecord attribute.

Table 5 HierarchyInfoRecord

<table>
<thead>
<tr>
<th>Component</th>
<th>Length (bits)</th>
<th>Offset (bits)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LID</td>
<td>16</td>
<td>0</td>
<td>For a switch: LID of switch port 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a CA or router: LID of the port.</td>
</tr>
<tr>
<td>PortNum</td>
<td>8</td>
<td>16</td>
<td>For a switch: port number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a CA or router: reserved.</td>
</tr>
<tr>
<td>HierarchyIndex</td>
<td>8</td>
<td>24</td>
<td>The specific hierarchy to which the included HierarchyInfo refers.</td>
</tr>
<tr>
<td>HierarchyInfo</td>
<td>512</td>
<td>32</td>
<td>HierarchyInfo contents; see Table 1 HierarchyInfo Attribute on page 16.</td>
</tr>
</tbody>
</table>
A15.7 HIERARCHY TEMPLATE FILE

The Hierarchy Template file is an XML file following a formal definition of an XML Schema. The key design considerations of the schema are described in this section followed by description of the key schema elements and the actual schema.

HI15-7: An endport supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, and programmed with specific hierarchy information shall be associated with a Hierarchy Template File with a TemplateGUID matching the node Hierarchy Information. That file shall be made available to the SM/SA and other management applications.

A15.7.1 KEY SCHEMA DESIGN CONSIDERATIONS

The following key considerations apply to the Template File:

- This specification imposes no assumptions or hard requirements on the format of the data stored in each HierarchyInfo:Level<N>Data component.
- Each hierarchy level HierarchyInfo:Level<N>Data may hold a different data set with a different format. Multiple different data sets may be defined and described by the schema, even for a single hierarchy.
- There is no requirement that the type of data is identical at a given level across all fabric ports.
- Some hierarchies may be deeper than others.
- Each HierarchyInfo:Level<N>Data component is decoded separately such that there is no information shared across levels.

A15.7.2 SCHEMA ELEMENTS

The hierarchy template schema includes at least one RecordSelector definition that is applied to several contiguous HierarchyInfo:Level<N>Data attribute components. Each one of the RecordSelector elements holds a field named RecordCode which maps to at least one Record element. Each Record element defines a set of fields included in one HierarchyInfo:Level<N>Data component. Each Record thus holds at least one Data field and exactly one Representation element describing how the Record is to be displayed. Each Data field holds a Bits element defining the specific bits of the HierarchyInfo:Level<N>Data attribute component making this field.

A15.7.2.1 BITS ELEMENT

A range of bits of the HierarchyInfo:Level<N>Data component defined as the [lsb:msb] range. Data elements use the "Bits" element to define which portion of the HierarchyInfo:Level<N>Data is used to provide their value.
The following example demonstrates declaration of a data field stored in bits 8 to 15 of the HierarchyInfo:Level<N>Data component:

```xml
<Bits lsb="8" msb="15" />
```

### A15.7.2.2 Schema Data Types

This section lists the schema elements which describe the types available for the fields which comprise the Level<N>Data components of the HierarchyInfo attribute.

#### A15.7.2.2.1 Integer Element

A data field that is interpreted as an integer (can be displayed in the record `Representation` using the decimal `%d`, unsigned decimal `%u` or hexadecimal `%x` format). The following example shows how an integer field named “SlotNum” which is stored in bits 8 to 15 of the HierarchyInfo:Level<N>Data component is defined:

```xml
<Integer name="SlotNum"> <Bits lsb="8" msb="15" /></Integer>
```

#### A15.7.2.2 Enum Element

An enumerated data field that maps the value stored in the field into a string (can be included in the record `Representation` using the `%s` format). For example an enumeration field named “BoardType” that maps the value 0 to the string “Leaf”, the value 1 to the string “Spine”, the value 2 to the string “Board” and the value 3 to the string “Reserved” is declared below:

```xml
<Enum name="BoardType"><Bits lsb="0" msb="1" /><EnumVal val="0" name="Leaf" /><EnumVal val="1" name="Spine" /><EnumVal val="2" name="Board" /><EnumVal val="3" name="Reserved" /></Enum>
```

#### A15.7.2.3 Representation Element

Defines how the information stored in data fields will be displayed, using a printf-like format. A format string and a list of data fields are provided as arguments. The following example for a `Representation` element defines a display format which uses a string to display the first argument `Data` field named “BoardType” and a decimal number for the second `Data` field named “SlotNum”.

```xml
<Representation format="%s%d" args="BoardType SlotNum" />
```

#### A15.7.2.4 Record Element

A specific data set stored in the HierarchyInfo:Level<N>Data (not including the RecordCode). Each `Record` must have a unique name. The following example shows a `Record` element holding two `Data` fields - “SlotNum” and “BoardType” - and a corresponding `Representation` element:

```xml
<Record name="Board"> <Integer name="SlotNum"> <Bits lsb="8" msb="15" /></Integer> <Enum name="BoardType">
```
<Bits lsb="0" msb="1" />
<EnumVal val="0" name="Leaf" />
<EnumVal val="1" name="Spine" />
<EnumVal val="2" name="Board" />
<EnumVal val="3" name="Reserved" />
</Enum>
<Representation format="%s%d" args="BoardType SlotNum" />
</Record>
A15.7.3 RECORDSELECTOR, RECORDCODE AND APPLYTOLEVELS ELEMENTS

The RecordSelector element references a set of Record elements that may be coded in the same set of HierarchyInfo:Level<N>Data levels.

The value stored in the bits of the RecordSelector element defines which one of these Record elements is actually stored in the Hierarchy-Info:Level<N>Data level. A set of RecordCode elements maps between the codes and their matching Record name. Another mandatory sub-field within the RecordSelector declaration named ApplyToLevels specifies the set of contiguous HierarchyInfo:Level<N>Data components to which this RecordSelector applies.

The example below declares two RecordSelector elements, one for HierarchyInfo:Level0Data, HierarchyInfo:Level1Data and Hierarchy-Info:Level2Data, and the other for the rest of the components HierarchyInfo:Level<N>Data. The first RecordSelector is stored at bit [31] and selects between two Record types named “Board” or “Device” while the second RecordSelector uses the set of bits to [28:31] and selects between three Record types: “Board”, “System” or “Rack”.

```xml
<RecordSelector>
  <ApplyToLevels min="0" max="2" />
  <Bits lsb="31" msb="31" />
  <RecordCode val="0" name="Board" />
  <RecordCode val="1" name="Device" />
</RecordSelector>
<RecordSelector>
  <ApplyToLevels min="3" max="12" />
  <Bits lsb="28" msb="31" />
  <RecordCode val="0" name="Board" />
  <RecordCode val="1" name="System" />
  <RecordCode val="2" name="Rack" />
</RecordSelector>
```

A special case of RecordSelector is when there is only one type of Record that may be coded in the set of levels specified by ApplyToLevels. In that case the Bits element msb and lsb fields should be set to 32 and a single RecordCode should be included. The val field of the RecordCode is reserved, thus it must be set to zero but is ignored. This coding enables using the entire 32 bits of the HierarchyInfo:Level<N>Data for storing Record data. The example below shows how such RecordSelector is defined:

```xml
<RecordSelector>
  <ApplyToLevels min="0" max="1" />
  <Bits lsb="32" msb="32" />
  <RecordCode val="0" name="SerialNum" />
</RecordSelector>
```

A15.7.4 FORMAL TEMPLATE FILE XML SCHEMA

The template file XML Schema is defined in Figure 4 Hierarchy Template File XML Schema on page 24.
Hierarchy Template File XML Schema

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="Level">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxExclusive value="13"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="ApplyToLevels">
    <xs:complexType>
      <xs:attribute name="max" type="Level" use="required" />
      <xs:attribute name="min" type="Level" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="Bit">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxExclusive value="32"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Bits">
    <xs:complexType>
      <xs:attribute name="msb" type="Bit" use="required" />
      <xs:attribute name="lsb" type="Bit" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="Enum">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Bits" minOccurs="1" maxOccurs="1" />
        <xs:element ref="EnumVal" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="EnumVal">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="val" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="Integer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Bits" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```
A15.7.5 A GENERIC PHYSICAL HIERARCHY TEMPLATE

In this section we provide an example template for Physical Hierarchy.

**Figure 5 Generic Physical Hierarchy Template**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<HierarchyTemplate GUID="0000000000000001" name="IBTA Physical Hierarchy">
  <RecordSelector>
    <ApplyToLevels min="0" max="12" />
    <Bits lsb="28" msb="31" />
    <RecordCode val="0" name="Device" />
    <RecordCode val="1" name="Board" />
    <RecordCode val="2" name="System" />
    <RecordCode val="3" name="Rack" />
    <RecordCode val="4" name="Room" />
    <RecordCode val="5" name="Campus" />
  </RecordSelector>
  <Record name="Device">
    <Integer name="SerialNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Representation format="%d" args="SerialNum" />
  </Record>
  <Record name="Board">
    <Integer name="SlotNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Enum name="BoardType">
      <Bits lsb="0" msb="1" />
      <EnumVal val="0" name="Leaf" />
      <EnumVal val="1" name="Spine" />
      <EnumVal val="2" name="Board" />
      <EnumVal val="3" name="Reserved" />
    </Enum>
    <Representation format="%s%d" args="BoardType SlotNum" />
  </Record>
  <Record name="System">
    <Integer name="TopUNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Enum name="SysType">
      <Bits lsb="0" msb="1" />
      <EnumVal val="0" name="SW" />
      <EnumVal val="1" name="H" />
      <EnumVal val="2" name="GW" />
      <EnumVal val="3" name="SYS" />
    </Enum>
    <Representation format="%s-U%d" args="SysType TopUNum" />
  </Record>
  <Record name="Rack">
    <Integer name="SerialNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Representation format="Rack%d" args="SerialNum" />
  </Record>
  <Record name="Room">
    <Integer name="SerialNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Representation format="Room%d" args="SerialNum" />
  </Record>
  <Record name="Campus">
    <Integer name="SerialNum"> <Bits lsb="8" msb="15" /> </Integer>
    <Representation format="Campus%d" args="SerialNum" />
  </Record>
</HierarchyTemplate>
```
Using the Generic Physical Hierarchy Template we provide here an example HierarchyInfo attribute query result for a node of the example hierarchy described in Figure 6 "Physical hierarchy view" which is a copy of Figure 3.

**Figure 6 Physical hierarchy view**

The SW2 located in SLOT3 of the switch system in U17 of rack 3 may return the following HierarchyInfo:

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TemplateGUID</td>
<td>0000000000000001h</td>
<td>GUID of the IBTA Physical Hierarchy Template</td>
</tr>
<tr>
<td>ActiveLevels</td>
<td>4</td>
<td>Device, Board, System and Rack</td>
</tr>
<tr>
<td>Level0Data</td>
<td>0000:0200h</td>
<td>Device, Num 2,</td>
</tr>
<tr>
<td>Level1Data</td>
<td>1000:0300h</td>
<td>Board, Slot 3, Leaf</td>
</tr>
<tr>
<td>Level2Data</td>
<td>2000:1100h</td>
<td>System, TopU 17, Switch</td>
</tr>
<tr>
<td>Level3Data</td>
<td>3000:0300h</td>
<td>Rack, Num 3</td>
</tr>
</tbody>
</table>

Software based on this template may display the name of the specific switch device as “Rack3/SW-U17/Leaf3/i2”.

The detailed procedure for decoding of Level1Data into “Leaf3” is:

- The value 1 stored in bits 31:28 which are the RecordSelector for all levels map the Level1Data to the “Board” Record
- The “Board” bits 15:8 define the slot number to be 3 and bits 1:0 defines the type to be a “Leaf”
- “Board” representation is of format %s%d which result in “Leaf3”
A15.7.6 A GENERIC PORT HIERARCHY TEMPLATE

The port hierarchy template example provides a means to track how device ports map to system ports. Only those ports which connect to the system panel may have such HierarchyInfo defined. The rest may return HierarchyInfo:ActiveLevels set to zero.

System ports may be arbitrarily named which represents a real challenge for a generic template. For this reason, we have coded several possible prefixes for the plug names and use them in the port name representation.
Figure 7  Generic Port Hierarchy Template

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<HierarchyTemplate GUID="0000000000000002" name="IBTA Generic Port Hierarchy">
  <RecordSelector>
    <ApplyToLevels min="0" max="0" />
    <Bits lsb="31" msb="31" />
    <RecordCode val="0" name="SlotBased" />
    <RecordCode val="1" name="Continuous" />
  </RecordSelector>
  <Record name="SlotBased">
    <Enum name="SlotPrefix">
      <Bits lsb="0" msb="1" />
      <EnumVal val="0" name="L" />  
      <EnumVal val="1" name="S" />
      <EnumVal val="2" name="Leaf" />
      <EnumVal val="3" name="Slot" />
    </Enum>
    <Enum name="PortPrefix">
      <Bits lsb="2" msb="3" />
      <EnumVal val="0" name="P" />
      <EnumVal val="1" name="J" />
      <EnumVal val="2" name="P" />
      <EnumVal val="3" name="Jack" />
    </Enum>
    <Enum name="Sep">
      <Bits lsb="4" msb="5" />
      <EnumVal val="0" name="." />
      <EnumVal val="1" name="/" />
      <EnumVal val="2" name="" />
    </Enum>
    <Integer name="port">
      <Bits lsb="8" msb="15" />
    </Integer>
    <Integer name="slot">
      <Bits lsb="16" msb="23" />
    </Integer>
    <Representation format="%s%d%s%d" args="SlotPrefix slot Sep PortPrefix port" />
  </Record>
  <Record name="Continuous">
    <Enum name="PortPrefix">
      <Bits lsb="0" msb="1" />
      <EnumVal val="0" name="P" />
      <EnumVal val="1" name="J" />
      <EnumVal val="2" name="Plug" />
      <EnumVal val="3" name="Jack" />
    </Enum>
    <Integer name="port">
      <Bits lsb="8" msb="23" />
    </Integer>
    <Representation format="%s%d" args="PortPrefix port" />
  </Record>
</HierarchyTemplate>
```
Using the above template port 2 of the device at Rack3/SW-U17/Leaf3/I2 may report back the following HierarchyInfo describing its port hierarchy information.

**Table 7 Port HierarchyInfo GetResponse Example**

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TemplateGUID</td>
<td>00000000000002h</td>
<td>GUID of the IBTA Port Hierarchy Template</td>
</tr>
<tr>
<td>ActiveLevels</td>
<td>1</td>
<td>System and Rack</td>
</tr>
<tr>
<td>Level0Data</td>
<td>0003:0220h</td>
<td>Slot based, slot 3, port 2, &quot;,&quot;, &quot;P&quot;, &quot;L&quot;</td>
</tr>
</tbody>
</table>

Decoding the above information with the Generic Port Hierarchy Template will result in the front panel of the switch system at jack L3/P1.
A15.8 COMPLIANCE

In order to claim compliance to the Hierarchy Information Management class a device shall meet all requirements specified in this section.

- **HI15-1:** A switch that sets PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one on its management port shall support the HierarchyInfo attribute on all its external ports. Page 14

- **HI15-2:** If the port supports Traps as indicated in the PortInfo:CapabilityMask.IsTrapSupported, and the port is capable of sending the enhanced (for other local changes) version of Trap 144 as indicated in the PortInfo:CapabilityMask.IsOtherLocalChangesNoticeSupported, and the port supports Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, then the SMA shall send a Trap 144 to the SM when the hierarchy information is modified by an entity on the sending port’s endnode at runtime Page 14

- **HI15-3:** If the management port supports Notices as indicated in the PortInfo:CapabilityMask.IsNoticeSupported, and the port is capable of logging the OtherLocalChanges notice as indicated in the PortInfo:CapabilityMask.IsOtherLocalChangesNoticeSupported, and the port supports Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one the SMA shall log a Notice using Notice:TrapNumber 144 when the hierarchy information is modified by an entity on the sending port's endnode at runtime Page 14

- **HI15-4:** An endport supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, shall respond to SubnGet(HierarchyInfo) and SubnSet(HierarchyInfo) according to the behavior described in the following section. Page 15

- **HI15-5:** An endport supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, shall track for every port the maximal HierarchyIndex of valid HierarchyInfo attributes and report it in the subsequent SubnGet(HierarchyInfo) MaxActiveIndex component Page 17

- **HI15-6:** If SA supports Hierarchy Information as indicated by setting its ClassPortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, it shall respond to SubnAdmGet() and SubnAdmGetTable() of the HierarchyInfoRecord attribute. Page 19

- **HI15-7:** An endport supporting Hierarchy Information as indicated by setting its PortInfo:CapabilityMask.IsHierarchyInfoSupported bit to one, and programmed with specific hierarchy information shall be associated with a Hierarchy Template File with a Tem-
plateGUID matching the node Hierarchy Information. That file shall be made available to the SM/SA and other management application. Page 20