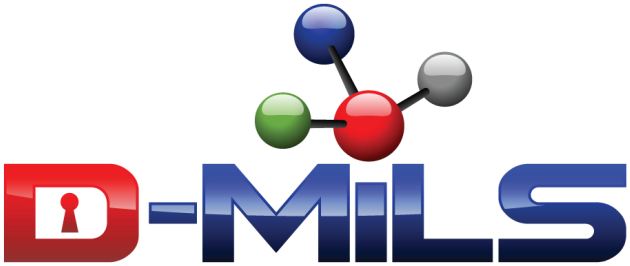

Project Number 318772

D5.3 Configuration correctness and semantics preserving transformations

Version 1.1
12 November 2015
Final

Public Distribution

The Open Group, Université Joseph Fourier, Inria

Project Partners: **Fondazione Bruno Kessler**, **fortiss**, **Frequentis**, **Inria**, **LynuxWorks**, **The Open Group**, **RWTH Aachen University**, **TTTech**, **Université Joseph Fourier**, **University of York**

Project Partner Contact Information

Fondazione Bruno Kessler Alessandro Cimatti Via Sommarive 18 38123 Trento, Italy Tel: +39 0461 314320 Fax: +39 0461 314591 E-mail: cimatti@fbk.eu	**fortiss** Harald Ruess Guerickestrasse 25 80805 Munich, Germany Tel: +49 89 36035 22 0 Fax: +49 89 36035 22 50 E-mail: ruess@fortiss.org
Frequentis Wolfgang Kampichler Innovationsstrasse 1 1100 Vienna, Austria Tel: +43 664 60 850 2775 Fax: +43 1 811 50 77 2775 E-mail: wolfgang.kampichler@frequentis.com	**LynuxWorks** Yuri Bakalov Rue Pierre Curie 38 78210 Saint-Cyr-l'Ecole, France Tel: +33 1 30 85 06 00 Fax: +33 1 30 85 06 06 E-mail: ybakalov@lnxw.com
RWTH Aachen University Joost-Pieter Katoen Ahornstrasse 55 D-52074 Aachen, Germany Tel: +49 241 8021200 Fax: +49 241 8022217 E-mail: katoen@cs.rwth-aachen.de	**The Open Group** Scott Hansen Avenue du Parc de Woluwe 56 1160 Brussels, Belgium Tel: +32 2 675 1136 Fax: +32 2 894 5845 E-mail: s.hansen@opengroup.org
TTTech Wilfried Steiner Schonbrunner Strasse 7 1040 Vienna, Austria Tel: +43 1 5853434 983 Fax: +43 1 585 65 38 5090 E-mail: wilfried.steiner@tttech.com	**Université Joseph Fourier** Saddek Bensalem Avenue de Vignate 2 38610 Gieres, France Tel: +33 4 56 52 03 71 Fax: +33 4 56 03 44 E-mail: saddek.bensalem@imag.fr
University of York Tim Kelly Deramore Lane York YO10 5GH, United Kingdom Tel: +44 1904 325477 Fax: +44 7976 889 545 E-mail: tim.kelly@cs.york.ac.uk	**Inria** Axel Legay Inria - Campus de Beaulieu 35042 Rennes, France Tel: +33 2 99 84 73 15 Fax: +33 2 99 84 71 71 E-mail: axel.legay@inria.fr

Contents

List of Figures

Document Control

Version	Status	Date
0.1	Doc outline, Exec Summary	25 January 2015
0.2	First version	5 February 2015
0.9	Mostly complete version	24 August 2015
1.0	Final version	25 September 2015
1.1	Finalized version	11 November 2015

Executive Summary

The D-MILS projects aims to build a methodology and a toolset for designing, developing and deploying MILS systems over a distributed platform. One of the tools provided by the D-MILS project, namely the *D-MILS Platform Configuration Compiler* (MPCC), automatically provides configuration file for the hardware and low-level software of the platform, accordingly to the system model. This document presents methods for checking that the configuration files produced by the MPCC are correct with respect to the source MILS-AADL model.

The MPCC takes as input a MILS-AADL specification of the application and the platform hardware. Based on the structure of the MILS-AADL specification and some annotations, the MPCC generates a global configuration of the platform. This configuration contains all the details for configuring the separations kernels, the MILS Network System and the network itself. Finally, for each system to configure, an *adapter* extracts the relevant information and generate the corresponding configuration file.

The goal of this document is to provide some methods to validate the output of the MPCC, that is the configuration files, with respect to its input, that is the MILS-AADL model and the model of the platform. First, we reconstruct a MILS-AADL model from the configuration files. Then, we compare the reconstructed MILS-AADL model with the original one, for one or several deployment constraints, encoded as relations between MILS-AADL models.

1 Introduction

Designing a distributed MILS (D-MILS) system yields a model, along with an assurance case arguing that the model meets the desired properties. The deployment of the system starts with the configuration of the platform. A correct configuration is crucial for meeting the security and safety goals of a D-MILS system. For ensuring security properties, intransitive non-interference should be enforced by the platform. In particular, two subjects that are not connected in the original model should not be able to communicate once deployed on the configured platform. For safety properties a low response time is necessary, which requires both sufficient computational resources and a low-latency network. In order to ensure that the global goals of the system are met, requirements on the configuration such that the ones above have to verified. The results of the verification yield arguments to enrich the assurance case.

As an outcome of the design phase, the system is expressed as a MILS-AADL model [1], which may include a basic model of the hardware. The MILS-AADL model hierarchy reflects the software architecture. More precisely, the main component of MILS-AADL model is a **`system`**. A **`system`** component is a composition of several **`system`** or **`subject`** components. A **`subject`** component describes a behavior and cannot contain another **`subject`** or a **`system`** subcomponent. When configuring the platform, we abstract the contents of the **`subject`** components, and see the system as a set of boxes (subjects), with some connections between them. This set of interconnected boxes correspond to the structure to be implemented by the configured platform. The configuration declares all the subjects and allocates appropriate resources for their execution and their intercommunication. Communication between subjects located on different nodes requires the configuration of the network. In the D-MILS project, the MILS platform configuration compiler (MPCC), or *configuration compiler* [2], automatically computes the mapping of subjects to nodes and generate the corresponding configuration.

This document provides some methods to check that configuration files produced by the configuration compiler are correct with respect to the original model and the hardware platform. These methods are not intended to be implemented in a separated tool, as an error in the configuration potentially comes from an error in the configuration compiler, which an user cannot correct. Rather, they provide an internal validation of the configuration, based on a transformation independent from the configuration process. In that sense, we increase the confidence that a validated configuration is correct, since a failure in both the configuration and the reconstruction process is less probable than a failure in the configuration process only.

In Section 2, we define the conventions that are used to encode hardware requirements and mapping constraints in the MILS-AADL model. In Section 3, we present how a MILS-AADL model can be reconstructed from the configuration files. Section 4 defines several relations for comparing the reconstructed MILS-AADL model with the original one. These relations do not depend on the hardware used, provided a reconstruction from the configuration file is provided. Finally, Section 5 links the verification methods proposed here with the assurance case.

2 Conventions in MILS-AADL

During the design phase, a MILS-AADL model is developed and validated through verification. In order to generate a configuration, the MILS-AADL model has to contain information about hardware, resources needed by each subject and mapping constraints. In this section we specify how to represent the following informations:

- the characteristics of the nodes in terms of processing power, memory, storage,
- the physical topology of the network,
- the required resources of each subject in terms of computing memory and storage, and
- the additional constraints for mapping subjects on the nodes.

2.1 Representing the hardware

In this subsection, we describe how the hardware is represented in a MILS-AADL model. This representation is mandatory for generating a configuration.

2.1.1 Representing Hardware of the Nodes

MILS-AADL features a dedicated catergory, **`node`**, in order to represent nodes. As for other components, a **`node`** type is declared by its interface and its implementation. The implementation of a node contains several subcompenents of category **`processor`**, **`memory`** or **`device`**. The characteristics of these components are specified to MPCC by adding annotations in the corresponding implementation. The folowing annotations are supported:

- `{MPCC: size(<size>)}` for **`memory implementation`**, where `<size>` indicates the size of the corresponding memory (in GB),
- `{MPCC: frequency(<freq>)}` for **`processor implementation`**, where `<freq>` indicates the frequency of the processor (for instance 2GHz),
- `{MPCC: family(<family>)}` for all kind of components, where `<family>` indicates the family of the annotated component implementation. Family is either `ram` or `disk` for memories, `cpu` for processors, and `ttcard` fo devices representing a Time Triggered Ethernet card.

Figure 1 shows an example where each of these annotations are used on **`processor`**, **`memory`** and **`device`** components. The declaration of a **`node implementation`** composes such components. For instance, the **`node implementation`** declared in Figure 2 models a computer with 4 cores (processors) running at 2 GHz, 16G of ram, 1T of hard drive space and a TTethernet network card. Several instances of such a node can then be included in the model, i.e. by including them into the main **`system`** declaration.

```
memory mem
end mem;

memory implementation mem.ram16G
  {MPCC: family(ram)}
  {MPCC: size(16)}
end mem.ram16G;

memory implementation mem.disk1T
  {MPCC: family(disk)}
  {MPCC: size(1204)}
end mem.disk1T;

processor proc
end proc;

processor implementation proc.intel2G
  {MPCC: frequency('2GHz')}
  {MPCC: schedule(100)}
end proc.intel2G

device ttcard
end ttcard;

device implementation ttcard.i
  {MPCC: family(ttcard)}
  {MPCC: schedule(100)}
end ttcard;
```

Figure 1: Hardware components declarations with annotations for the MPCC

```
node dell
end dell;

node implementation dell.i
  subcomponents
    cpu0: processor proc.intel2G;
    cpu1: processor proc.intel2G;
    cpu2: processor proc.intel2G;
    cpu3: processor proc.intel2G;

    ram:  memory mem.ram16G;
    disk: memory mem.disk1T;
    es: device ttcard.i ;
end dell.i;
```

Figure 2: Declaration of a node type reusing components defined in Figure 1

2.1.2 Representing Hardware of the Network

To complete the hardware description of the platform, we need to describe the physical aspects of the network. Network-related hardware comprises switches, network cards and cables. Cards are already represented as a part of the node. Switches are represented by components of category **bus**.

We represent the connectivity between nodes and switches by using the **accesses** keyword. These keywords are added at the declaration of each node, to specify to which switch they are connected.

The cabling is detailed by a set of annotations, one for each cable. The content of an annotation has the following syntax:
`phylink (` *port_ref*`,` *port_ref*`,` *attributes* `)`
Where

- *port_ref* is a port reference, of the form `[`*node*`,`*device*`,`*port*`]` or `[`*switch*`,`*port*`]` and
- *attributes* is of the form `[ media(<media>), length(<length>)]` where `<media>` is either `copper` or `fiber`, and `<length>` indicates the length of the cable (in m).

The type and length of the cables is taken into account by the TTTech tools to compute the propagation time of the information on the cables and adjust the schedule accordingly. In particular, transmitting information on very long copper cables can cause a non-negligible delay.

Figure 3 depicts how network-related hardware is described in MILS-AADL. The annotation in the example states that the port 1 of the card in `tt1` is connected to the port 1 of the switch, and port 1 of the card in `tt2` is connected to the port 2 of the switch.

```
bus switch
end switch;

bus implementation switch.i
end switch.i;

system implementation main.i
  {MPCC: phylink( pr({tt1},es,p1), pr({sw0},p1),
                  [length(2m),media(copper)] )}
  {MPCC: phylink( pr({tt2},es,p1), pr({sw0},p2),
                  [length(2m),media(copper)] )}
  subcomponents
    sw0: bus switch.i;
    tt1: node dell.i accesses sw0;
    tt2: node dell.i accesses sw0;
    [...]
end main.i;
```

Figure 3: Description of network-related hardware

2.2 Representing Deployment Constraints

In this subsection, we detail how deployment constraints are represented in the MILS-AADL model. These constraints include the minimal hardware requirements for each subject to run properly. Additional constraints on the repartition of the subjects to the nodes are also considered.

2.2.1 Representing the Hardware Requirements

Each subject needs at least RAM and probably disk storage to function properly. Such requirements can be expressed by adding annotations in the declaration of the **`subject implementation`**. Each annotation requires the allocation of a particular resource to the subject being declared. These annotations have the following form:
`{MPCC: <category>(_, [<characteristics>] ) }`
where

- `<category>` corresponds to the type of resource needed (i.e. **`processor`**, **`memory`** or **`device`**
- `<characteristics>` details the resource (i.e. type, size ...) using the same attributes as for the description of the hardware.

Figure 4 show an example of these annotations. In that example, the subject "low" needs 2G of RAM and 10G of hard drive space.

```
subject implementation low.i
  {MPCC: memory(_, [family(ram), size(2)] ) }
  {MPCC: memory(_, [family(disk), size(10)] ) }
 [...]
end low.i
```

Figure 4: Hardware constraints associated to a subject.

2.2.2 Representing Mapping Constraints

Mapping constraints define how the subjects can be deployed to the nodes. For instance, one can impose that redundant subjects are deployed to different nodes, in order to obtain a system that survives the failure of a node. We propose several primitives to define mapping constraints. In the following, s_1 is a subject, $S = [s_1, \ldots, s_n]$ is a list of subjects and $N = [n_1, \ldots, n_k]$ is a list of nodes.

- `same(S)` - all the subjects in the list S are allocated to the same node
- `in(`s_1`,`N`)` - subject s_1 is allocated to one of the nodes in list N
- `subset(`S`,`N`)` - the subjects in the list are allocated to the nodes in the list N
- `not_same(S)` - the subjects in the list S are not all allocated to the same node
- `not_in(`s_1`,`N`)` - subject s_1 is allocated to a node that is not in list N
- `not_subset(`S`,`N`)` - the subjects in the list are not allocated to the nodes in the list N
- `all_different(`S`)` - subjects in list S are all allocated to different node

These mapping constraints are defined in annotations to the **`system implementation`** that contains the nodes and subjects involved in the constraints. Annotations have the form:
`{MPCC: deployment(<primitive>)}`
where `<primitive>` is one of the primitives defined above. Figure 5 shows an example of such annotations.

```
system implementation Sys.impl
  {MPCC: deployment(not_same([low,user])) }
  {MPCC: deployment(same([dispatch,user])) }
  subcomponents
    high : subject Hsubject; -- high security network
    low :  subject Lsubject; -- low security network
    dispatch:  subject Dsubject;
    user:  subject Usubject;
   [...]
end Sys.impl;
```

Figure 5: Mapping constraints: subjects `low` and `user` should not be deployed on the same node, but `dispatch` and `user` should be on the same node.

For an example of a complete MILS-AADL specification, see Appendix A.

3 Reconstructing a MILS-AADL Model from the Configuration Files

This section explains how to reconstruct a MILS-AADL model from the configuration files. The purpose of this transformation is to represent the configuration files in a format that can be compared with the source MILS-AADL model, in order to check that the configuration implements the source model.

For the aforementioned check to be possible, the reconstructed model has to include sufficient information. A configuration is basically an assignment of resources to subjects. Assigned resources comprise computational resources (such as processors, memory) and resources for communication (communication ports). Therefore, the reconstructed model shall include information about the resources allocated to each subject, as well as their interconnection through the communication ports.

In order to allow for a more modular implementation, we decompose the reconstruction of the original model in two steps. The first step consists in building a configuration normal form from the configuration files of the components. The second step consists in reconstructing a MILS-AADL model from the configuration normal form. Note that the first step depends on the hardware combination used to implement the D-MILS platform, whereas the second step is independent of a particular hardware. In this section, we focus on the second step.

3.1 Structure of the Reconstructed Model

The reconstructed model depicts the choices made during the configuration process. We compare it to the MILS-AADL model including the constraints, to check that the configuration meets the constraints. More precisely, the reconstructed model should allow us to check:

- the intransitive non-interference: the communication between two subjects is allowed in the reconstructed model if and only if it is allowed in the original model,
- the hardware constraints: each subject has enough resources to function properly,
- the mapping constraints: the subjects are correctly deployed to the nodes.

In order to capture these elements, we structure the reconstructed model according to the physical layout of the system. We first detail the higher-level point of view, where the system is a seen as a set of nodes. Then we detail how each node is described as a collection of resources and subjects. The hierarchy of categories that can be included in a reconstructed model are depicted in Figure 6.

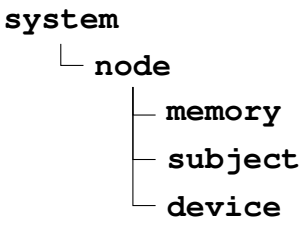

Figure 6: Hierarchy of categories allowed in the reconstructed model

Main system declaration At the higher level, the system is a set of nodes that communicate through dedicated channels. In terms of MILS-AADL, the main **`system`** interface declaration is empty or contains only ports representing communication with the environment of the system. The main **`system implementation`** can only declare subcomponents of category **`node`**, each one representing one of the hardware node composing the platform. Communication channels between nodes are represented by event port connections between the ports of the nodes. We use event ports because the type of data conveyed by the channel is irrelevant for analyzing the intransitive non-interference.

Declaring a node A **`node`** contains the subjects that have been deployed on it. It also includes a set of resources (ram, disk, devices) modeled as **`memory`** and **`device`** components. We use the **`accesses`** keyword to depict the fact that a subject can access a given resource. Two types of links appear in the **`connections`**: links between subjects that are hosted locally and links between a local subject port and an outer port of the node. As in the main **`system`** declaration, these links are modelled by event ports connections.

3.2 Implementing the Reconstruction

The reconstruction of a MILS-AADL model from a set of configuration files is implemented in two steps. First, a Mils Configuration Normal Form is constructed from the configurations files. Then we use a special adapter of the MPCC to construct a MILS-AADL model from the normal form.

Adapters are modules from the MPCC whose role is to project the configuration normal form onto a configuration for a part of the system. The generation of separation kernel configuration files, global information flow policy and network configuration file is done by dedicated adapters. In order to target a new hardware or separation kernel for the D-MILS platform, it is sufficient to write the corresponding adapter. In this subsection, we focus mainly on the second part of the reconstruction.

From configuration files to configuration normal form. This part is dependent on the technologies used to implement the D-MILS platform. We do not detail here this transformation but rather give the main steps:

1. Check the consistency between the different configuration files. For instance, check that communication channel endpoints defined in the separation kernel configuration correspond to channels defined in the network configuration.
2. Extract the set of subjects and their resources from the separation kernel configuration.
3. Construct the communication between subjects from the separation kernel and network configuration.

This reconstruction part is more complex than applying the inverse of the adapters. Indeed, it needs to combine the inverse of several adapters in order to form a coherent configuration normal form.

Examples of configuration files for the current version of the D-MILS platform can be found in Appendices C, D and E.

From the configuration normal form to MILS-AADL This transformation is implemented as an adapter. It exports the information contained in the configuration normal form to a MILS-AADL model. The reconstructed model has the structure defined in Subsection 3.1.

The configuration normal form contains all the information needed to generate the reconstructed MILS-AADL model. Figure 7 presents a human-readable version of a configuration normal form. This version exhibits a structure similar to the one from Subsection 3.1. At the top-level, each node is listed, here the configuration contains two nodes (tt1 and tt2). For each node, the subjects running on it are listed (`asubject`) as well as their memory resources (`amemobj`). Additional subjects (`xsubject`) and their resources (`xmemobj`) represent the resources needed for the separation kernel and MILS Network Subsystem. For each subject, the configuration lists outgoing connections to another subject on the same node (`ssconn`), to an object on the same node (`soconn`) and to a subject on another node (`xconn`).

In order to obtain a MILS-AADL model that conforms to Subsection 3.1, the adapter works in a bottom up fashion.

First, **`memory`** objects are created, with their corresponding types and sizes written in annotation. Types and sizes do not appear on Figure 7, but they are part of the configuration normal form. Similarly, **`subject`** components are created, with an empty implementation. The interface of subjects initially contains only outgoing ports, named after the origin port of their outgoing connection.

Each **`node`** component is defined by adding as subcomponents the previously **`memory`** and **`subject`** components. The name of the subcomponents is the name used in the configuration normal form, i.e. the parameter of `asubject()`. During this definition, connections from the configuration normal form are treated as follows:

- `soconn` (connection from subjects to objects) are encoded with the **`accesses`** keyword in the declaration of the subject instance
- `ssconn` (connection between subjects of the same node) are encoded by
  1. declaring an ingoing port in the destination subject, named after the destination port of the connection and
  2. declaring a MILS-AADL connection between the ports involved in the connection, in the **`connections`** section of the **`node implementation`**.
- `xconn` (connection to a subject on another node). Such a connection involves 4 elements: the origin port in the local subject, the origin RAO in the local node, the destination RAO of the remote node and the destination port in the remote subject. It is encoded by:
  1. declaring an outgoing port in the node interface, named after the origin RAO declared in the configuration and
  2. declaring a MILS-AADL connection between the origin port of the local subject and the newly declared outgoing port, in the **`connections`** section of the **`node implementation`**.

Finally, the main system implementation is declared by having the newly created node as subcomponents. Connections between subject on different nodes (`xconn`) are completed by:

```
node(tt1)
    asubject(low)
      soconn: low -> low_ram, soconn(low__low_ram)
      soconn: low -> low_disk, soconn(low__low_disk)
      ssconn: low -> high, ssinfo(resL)
    asubject(high)
      soconn: high -> high_ram, soconn(high__high_ram)
      soconn: high -> high_disk, soconn(high__high_disk)
      xconn: from tt1:high -> tt1:rao_res(mns) -> tt2:rao_res
                                                  -> tt2:dispatch xconn(res)
    xsubject(mns)
      soconn: mns -> mns_ram, soconn(mns__mns_ram)
      soconn: mns -> mns_disk, soconn(mns__mns_disk)
    amemobj(low_ram)
    amemobj(low_disk)
    amemobj(high_ram)
    amemobj(high_disk)
    xmemobj(mns_ram)
    xmemobj(mns_disk)
node(tt2)
    asubject(user)
      soconn: user -> user_ram, soconn(user__user_ram)
      soconn: user -> user_disk, soconn(user__user_disk)
      ssconn: user -> dispatch, ssinfo(cmd)
      ssconn: user -> dispatch, ssinfo(switchH)
      ssconn: user -> dispatch, ssinfo(switchL)
    asubject(dispatch)
      soconn: dispatch -> dispatch_ram, soconn(dispatch__dispatch_ram)
      soconn: dispatch -> dispatch_disk, soconn(dispatch__dispatch_disk)
      ssconn: dispatch -> user, ssinfo(display)
      xconn: from tt2:dispatch -> tt2:rao_cmdH(mns) -> tt1:rao_cmdH
                                                    -> tt1:high xconn(cmdH)
      xconn: from tt2:dispatch -> tt2:rao_cmdL(mns) -> tt1:rao_cmdL
                                                     -> tt1:low xconn(cmdL)
    xsubject(mns)
      soconn: mns -> mns_ram, soconn(mns__mns_ram)
      soconn: mns -> mns_disk, soconn(mns__mns_disk)
    amemobj(user_ram)
    amemobj(user_disk)
    amemobj(dispatch_ram)
    amemobj(dispatch_disk)
    xmemobj(mns_ram)
    xmemobj(mns_disk)
```

Figure 7: Textual representation of the configuration normal form

1. creating an ingoing port in the target node, named after the destination RAO in the configuration normal form,
2. creating an ingoing port in the target subject, named after the destination port in the configuration normal form,
3. adding a MILS-AADL connection in the destination node declaration, between the destination RAO port and the destination subject port, and
4. adding a MILS-AADL connection in the main system declaration, between the origin RAO port and the destination RAO port.

Appendix B presents the MILS-AADL model obtained by reconstructing the configuration obtain for the specification in Appendix A.

4 Verification of the Configuration

At this point, we have two different MILS-AADL models. On one hand, there is the original MILS-AADL model that was fed to the MPCC in order to produce some configuration files. On the other hand, there is a structural MILS-AADL model, with one level of hierarchy, that represents the configuration implemented in the configuration files. In order to check whether the configuration is correct with respect to the original model, we define several relations between MILS-AADL models. Each relation is intended to verify a particular aspect of the correctness. Depending on the focus of the verification, different sets of relation can be used to perform the check.

We present different relations that can be used to check the configuration. The relations defined below are not restricted to the current technology used in the D-MILS project. These definitions can be used with any other technology for the separation kernel and the network, provided a reconstruction from their configuration to a MILS-AADL model can be provided.

4.1 Checking Platform-level Intransitive Non-Interference

This relation is an equivalence relation that aims to check that both MILS-AADL models considered allow the same communication between the subjects. In other words, we check that, when abstracting the behaviour of the subjects, both models have the same non-interference properties.

We say that a subject instance can send a message to another subject instance if there exists an event or event data connection or a flow from the first subject to the other subject. Note that subjects can be composed inside a system or a node component, in which case there is no direct connections between the two subjects. Therefore we first define this notion on ports (here port refers to the port of a component instance). We say that a port p_1 is connected to a port p_2 if it satisfies the following recursive property:

- p_1 and p_2 are the same port of a **`system`** or a **`node`** component, and is connected both from the inside and the outside of that component ,

- there is a declared connection or flow from p_1 to p in the **`system`** implementation of the **`system`** instance that involve both p_1 and p, and p is either p_2 or connected to p_2

We say that a subject instance s_1 can send a message to a subject instance s_2 iff there exist a ports p_1 of s_1 and a port p_2 of s_2 such that p_1 is connected to p_2.

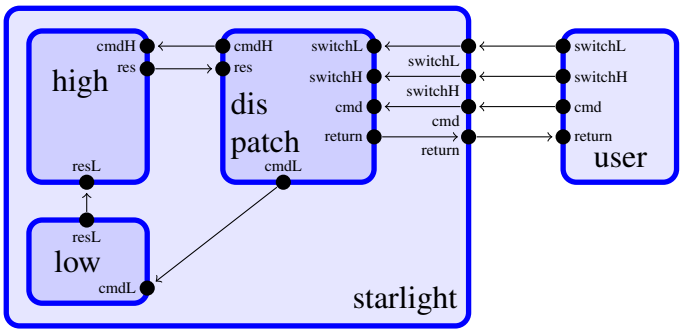

Figure 8: Graphical representation of the original MILS-AADL model

For instance, in Figure 8, the subject instance named `dispatch` can send a message to `user`, even though they are not directly connected. However, `low` cannot send a message to `dispatch`, the connection between them allows only communication from `dispatch` to `low`.

Given a subject s in a model M, we denote by $REACH_M(s)$ the names of subjects to which it can send a message. Given a model M, we denote by $Subjects(M)$ the set of names of subject declared in M. We say that two MILS-AADL models are related for the Platform-level Intransitive Non-Interference, denoted $A \sim_{PINI} B$ iff $Subjects(A) = Subjects(B) \wedge \forall s \in Subjects(A)\ \ REACH_A(s) = REACH_B(s)$. That is, two models are equivalent if the allowed communications are the same in both models.

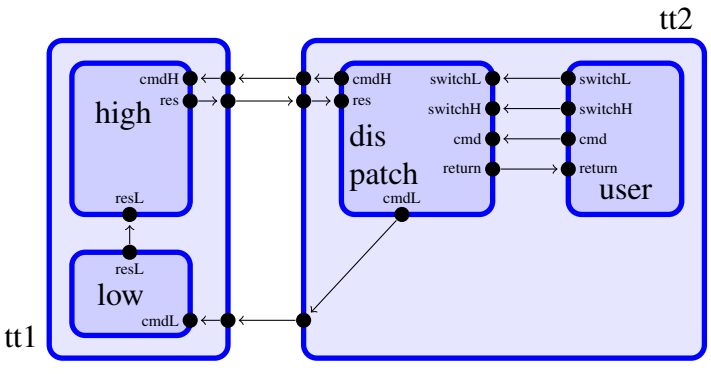

Figure 9: Graphical representation of the MILS-AADL model reconstructed from the normal form configuration depicted in Figure 7

The models depicted in Figure 8 and Figure 9 verify this relation. The reader may check it by building for each subject the set $REACH$ in both model and verify equality of both sets.

4.2 Checking Hardware Resources

This relation aims to ensure that the resources allocated to a subject are sufficient for its execution. In the original model, annotations attached to a subject specify how much resources are required for the subject to execute correctly, as explained in Section 2. In the reconstructed model, each subject can access a given set of resources, indicated through the **`accesses`** keyword.

We start by specifying how to check that enough random access memory is available for the subjects. Given a subject instance s, we denote by $RAM_M(s)$ the amount of random access memory specified or allocated to the subject s. In the original model A, the value of $RAM_A(s)$ is obtained by reading the annotations associated to the **`subject implementation`** of s. In the reconstructed model B, the value of $RAM_A(s)$ is obtained by reading the annotations associated to the **`memory implementation`** of the memory that s can access. This relation is not an equivalence relation as it is not symmetric. We say the reconstructed model B provides enough memory resource with respect to the original model A, denoted $A \leq_{RAM} B$, iff $Subjects(A) = Subjects(B) \wedge \forall s \in Subjects(A) \quad RAM_A(s) \leq RAM_B(s)$. The relation simply states that amount of memory allocated to the subject in the configuration is bigger than the requested amount in the original model. A similar relation, $\leq_{DISK}$ is constructed for checking that the subject have enough disk storage.

Some subject might require some specific hardware to function properly, such as a graphics adapter for a subject in charge of displaying information. For each family F of hardware devices, we define $F_M(s)$ as the number of devices of family F needed or allocated to the subject instance s in the model M. As previously, we define the relation $\leq_F$ the original model A and the reconstructed model B as $Subjects(A) = Subjects(B) \wedge \forall s \in Subjects(A) \quad F_A(s) \leq F_B(s)$.

Finally, we need to check that each node does not allocate more resources that it actually has. We denote by $\mathcal{R}$ a type of resource, which can be RAM, $DISK$ or a particular hardware family F. Given a node n, we denote by $Total\mathcal{R}_M(n)$ the amount or number of resource $\mathcal{R}$ available on or allocated by the node n in the model M. The characteristics of the node's hardware are specified in the original model A. In the reconstructed model B, $Total\mathcal{R}_B(n)$ denotes the quantity of resource $\mathcal{R}$ allocated to the subjects deployed on n The reconstructed model specifies the mapping by including subjects mapped to a particular node as subcomponents of that node. Given a subject instance s of a model M, we denote by $f_M(s)$ the father of s in the model M, that is the component that includes s as a subcomponent. Formally, $TotalRAM_B(n) = \sum_{f_B(s)=n} RAM_B(s)$. We write $A \geq_{Total\mathcal{R}} B$ if for each node n, $Total\mathcal{R}_A(n) \geq Total\mathcal{R}_A(n)$.

A reconstructed model B satisfies the hardware resources specified the original model A if A and B satisfy the relations $\leq_{\mathcal{R}}$, and $\geq_{Total\mathcal{R}}$ for all types of resources $\mathcal{R}$. This relation is verified if each subjects gets enough resources and each node does not allocate more resources than it has.

4.3 Checking Mapping Constaints

This relation verifies that the mapping constraints included in the original model are satisfied by the reconstructed model. We reuse $f_M(s)$ to define the father of s in the model M, that is the component that includes s as a subcomponent.

We denote $C(A)$ the set of mapping constraints included in the original model A. Given a constraint c in C, and a reconstructed model B, we write $B \models c$ if B satisfy the constraint c. The formal definition of $B \models c$ is given below, where s_1 and s_2 are subjects, S is a set of subjects and N is a set of nodes.

- $B \models$`same(`s_1`,`s_2`)` iff $f_B(s_1) = f_B(s_2)$: s_1 and s_2 are mapped to the same node.
- $B \models$`in(`s_1`,`N`)` iff $\exists n \in N\ f_B(s_1) = n$ s_1 is mapped to a node that is in the list N.
- $B \models$`subset(`S`,`N`)` iff $\forall s \in S\ B \models$ `in(`s_1`,`N`)` every subject in the list S is mapped to a node that is in the list N.
- $B \models$`not_same(`s_1`,`s_2`)` iff $B \not\models$`same(`s_1`,`s_2`)`
- $B \models$`not_in(`s_1`,`N`)` iff $B \not\models$`in(`s_1`,`N`)`
- $B \models$`not_subset(`S`,`N`)` iff $B \not\models$`subset(`S`,`N`)`
- $B \models$`all_different(`S`)` iff $\neg\exists s_1, s_2 \in S\ s_1 \neq s_2 \wedge f_B(s_1) = f_B(s_2)$: there is no pair of distinct subjects from the list S that are mapped to the same node..

We say that the reconstructed model B satisfies the constraints from the original model A, denoted $B \models C(A)$ if all the constraints from A are satisfied by B: $\forall c \in C(A) B \models c$.

4.4 Implementation

Depending on the context, checking that all the above relations hold might not be necessary. Furthermore, some additional relation between the original and the reconstructed model might be of interest. Therefore in order to provide a modular and extensible verification of the reconstructed model with respect to the original model, we implement each of these relation as simple (python) functions. The functions take the representations of both the original and the reconstructed model as arguments. They return a tuple containing a boolean and a string. The boolean indicates whether the two models satisfy the relation to check and the string explains why the relation doesn't hold if the boolean is false.

These functions are called by a main python script that takes in argument:

- the files containing the original model
- the file containing the reconstructed model
- the list of relations to check

The python script parses the two models using the MILS-AADL parser. It then sequentially calls the functions corresponding to the relations to check. If one of them evaluates to false, it returns the message to the user stating why the corresponding relation does not hold. Otherwise, the script reports that the reconstructed model is correct with respect to the original model for the relations given in argument.

Checking Platform-level Intransitive Non-Interference The function loops over the subjects. For each subject, it builds the set of subject to which a message can be sent, in both the original and the reconstructed model s. If the two set matches, the loop continues. Otherwise, the function returns false, with a message indicating the error, for instance. "The subject s can reach the subject s' in the original model but not in the reconstructed model."

Checking Hardware Resources Allocation The function first loops over subjects and checks that the required value for each resource in the original model is met in the reconstructed model. If it is not the case, the function returns false, with a message similar to "The subject s has not enough resource of type $\mathcal{R}$: needed X, allocated Y". Then the function loops over the nodes and checks that each of them do not allocate more resource than it as. If a node allocates more resources than it has, the function returns false with an appropriate error message.

Checking Mapping Constraints The function loops over the constraints defined in the original model and checks whether they hold in the reconstructed model. If one of them is not met, the function returns false with an appropriate error message.

5 Assurance Considerations

The methods presented here provide some arguments to assure the platform is correctly configured, with respect to different criteria. In particular, the different relations provided in Section 4 between the reconstructed model and the original model allow a selective verification of different correctness notion for the configuration.

Some of these relations do not require further verification. For instance, when checking that sufficient resources have been allocated to each subject, the result is not conditioned by any other verification. In that case, the verification process for the configuration can be considered as an argument.

Some other relations are not sufficient to ensure that the system is properly configured, but do require further verifications at the level of the subjects. For instance, consider two subjects $s1$ and $s2$, such that there are two different connections from $s1$ to $s2$ and no backwards connection in the original model. Depending on the situation, several configurations are possible for that situation. If $s1$ is trusted, it might be sufficient to have a bidirectional connection between the two subjects and let $s1$ schedule the emission of messages and firewall all incoming message from $s2$. If $s1$ is not trusted and the two connections between $s1$ and $s2$ do not have the same priority and volume, for instance one correspond to a braking command while the other is a mp3 stream, then one might rely on the network configuration to ensure that braking commands always arrive in time.

In the first case, we require an additional verification on $s1$, that is that the scheduling of outgoing messages to $s2$ is correct. This has to appear in the assurance case.

In the second case, we do not need further verification on subject $s1$ but need to check a tighter equivalence between the original and the reconstructed model. In particular, the information about the communication rates in both links from $s1$ to $s2$ has to be checked.

In both cases, the subject is presented a given set of virtual cards, each of them corresponding to one (case 2) or more (case 1) ports. A further needed point is then that the subject application sends the messages through the corresponding card. To illustrate, consider that the subject is implemented as a GNU/Linux operating system. In that case, the RAOs are implemented as virtual Ethernet cards, and the order in which they are declared in the configuration file correspond to the order in which network interfaces are numbered by the Linux kernel.

Acronyms

D-MILS distributed MILS 2

MPCC MILS platform configuration compiler 2

References

[1] Specification of MILS-AADL. Technical Report D2.1, Version 2.0, D-MILS Project, July 2014. http://www.d-mils.org/page/results. 2

[2] Distributed MILS Platform Configuration Compiler. Technical Report D5.2, Version 0.2, D-MILS Project, March 2014. http://www.d-mils.org/page/results. 2

A MILS-AADL Original Model

This model contains the full version of the starlight model passed to the configuration compiler.

```
-------------------------------------------------------------------------------
-- Hardware components
-------------------------------------------------------------------------------

memory mem
end mem;

memory implementation mem.ram16G
  {MPCC: family(ram)}
  {MPCC: size(16)}
end mem.ram16G;

memory implementation mem.disk1T
  {MPCC: family(disk)}
  {MPCC: size(1204)}
end mem.disk1T;

processor proc
end proc;

processor implementation proc.intel2G
  {MPCC: frequency('2GHz')}
  {MPCC: schedule(100)}
end proc.intel2G;

device ttcard
end ttcard;

device implementation ttcard.i
  {MPCC: family(ttcard)}
  {MPCC: schedule(100)}
end ttcard.i;

bus switch
end switch;

bus implementation switch.i
end switch.i;


-------------------------------------------------------------------------------
-- Description of a node
-------------------------------------------------------------------------------

node dell
end dell;

node implementation dell.i
  subcomponents
```

```
    cpu0: processor proc.intel2G;
    cpu1: processor proc.intel2G;
    cpu2: processor proc.intel2G;
    cpu3: processor proc.intel2G;

    ram:  memory mem.ram16G;
    disk: memory mem.disk1T;
    es: device ttcard.i ;
end dell.i;


-------------------------------------------------------------------------------
-- Software components (subjects and systems composing subjects)
-------------------------------------------------------------------------------

subject Hsubject
  features
    cmdH: in data port int ;
    resL: in data port int ;
    res: out data port int ;
end Hsubject;

subject implementation Hsubject.i
  {MPCC: memory(_, [family(ram), size(2)] ) }
  {MPCC: memory(_, [family(disk), size(10)] ) }
 -- we don't reproduce the complete implementation here
end Hsubject.i;


subject Lsubject
  features
    cmdL: in data port int ;
    resL: out data port int ;
end Lsubject;

subject implementation Lsubject.i
  {MPCC: memory(_, [family(ram), size(2)] ) }
  {MPCC: memory(_, [family(disk), size(10)] ) }
 -- we don't reproduce the complete implementation here
end Lsubject.i;


subject Dsubject
  features
    cmdH: out data port int ;
    cmdL: out data port int ;
    res: in data port int ;
    switchL: in event port;
    switchH: in event port;
    cmd: in data port int;
    return: out data port int;
end Dsubject;
```

```
subject implementation Dsubject.i
  {MPCC: memory(_, [family(ram), size(4)] ) }
  {MPCC: memory(_, [family(disk), size(20)] ) }
 -- we don't reproduce the complete implementation here
end Dsubject.i;


subject Usubject
  features
    switchL: out event port;
    switchH: out event port;
    cmd: out data port int;
    return: in data port int;
end Usubject;

subject implementation Usubject.i
  {MPCC: memory(_, [family(ram), size(1)] ) }
  {MPCC: memory(_, [family(disk), size(10)] ) }
 -- we don't reproduce the complete implementation here
end Usubject.i;

system Starlight
  features
    switchL: in event port;
    switchH: in event port;
    cmd: in data port int;
    return: out data port int;
end Starlight;

system implementation Starlight.i
 subcomponents
    high : subject Hsubject;
    low :  subject Lsubject;
    dispatch:  subject Dsubject;
 connections
    port cmd -> dispatch.cmd ;
    port switchH -> dispatch.switchH ;
    port switchL -> dispatch.switchL ;
    port dispatch.return -> return ;
    port dispatch.cmdL -> low.cmdL ;
    port dispatch.cmdH -> high.cmdH ;
    port low.resL -> high.resL ;
    port high.res -> dispatch.res ;
end Starlight.i;


-------------------------------------------------------------------------------
-- Main
-------------------------------------------------------------------------------

system main
end main;
```

```
system implementation main.i
  {MPCC: phylink( pr(tt1,es,p1), pr(sw0,p1),
                  [length('2m'),media(copper)] )}
  {MPCC: phylink( pr(tt2,es,p1), pr(sw0,p2),
                  [length('2m'),media(copper)] )}
  {MPCC: deployment(not_same([starlight_low,user])) }
  {MPCC: deployment(same([starlight_dispatch,user])) }

  subcomponents
    sw0: bus switch.i;
    tt1: node dell.i accesses sw0;
    tt2: node dell.i accesses sw0;
    starlight: system Starlight;
    user:  subject Usubject;
 connections
    port user.cmd -> starlight.cmd ;
    port user.switchH -> starlight.switchH ;
    port user.switchL -> starlight.switchL ;
    port starlight.return -> user.return ;
end main.i;
```

B MILS-AADL Reconstructed Model

This model is reconstructed from the configuration obtained from the model in Appendix A.

```
-------------------------------------------------------------------------------
-- Declaration of memory segments
-------------------------------------------------------------------------------
memory m_low_disk
end    m_low_disk;

memory implementation m_low_disk.i
  {MPCC: size(10)}
  {MPCC: family(disk)}
end    m_low_disk.i;

memory m_low_ram
end    m_low_ram;

memory implementation m_low_ram.i
  {MPCC: size(2)}
  {MPCC: family(ram)}
end    m_low_ram.i;

memory m_high_disk
end    m_high_disk;

memory implementation m_high_disk.i
  {MPCC: size(10)}
  {MPCC: family(disk)}
end    m_high_disk.i;

memory m_high_ram
end    m_high_ram;

memory implementation m_high_ram.i
  {MPCC: size(2)}
  {MPCC: family(ram)}
end    m_high_ram.i;

memory m_dispatch_disk
end    m_dispatch_disk;

memory implementation m_dispatch_disk.i
  {MPCC: size(20)}
  {MPCC: family(disk)}
end    m_dispatch_disk.i;

memory m_dispatch_ram
end    m_dispatch_ram;

memory implementation m_dispatch_ram.i
  {MPCC: size(4)}
  {MPCC: family(ram)}
```

```
end     m_dispatch_ram.i;

memory m_user_disk
end     m_user_disk;

memory implementation m_user_disk.i
  {MPCC: size(10)}
  {MPCC: family(disk)}
end     m_user_disk.i;

memory m_user_ram
end     m_user_ram;

memory implementation m_user_ram.i
  {MPCC: size(1)}
  {MPCC: family(ram)}
end     m_user_ram.i;

memory m_mns_ram
end     m_mns_ram;

memory implementation m_mns_ram.i
  {MPCC: size(4)}
  {MPCC: family(ram)}
end     m_mns_ram.i;

-------------------------------------------------------------------------------
-- Declaration of subjects
-------------------------------------------------------------------------------


subject s_high
  features
    res: out event port   ;
    cmdH: in event port   ;
    resL: in event port   ;
end s_high;

subject implementation s_high.i
end s_high.i;

subject s_low
  features
    resL: out event port ;
    cmdL: in event port ;
end s_low;

subject implementation s_low.i
end s_low.i;

subject s_dispatch
  features
    cmdH: out event port ;
```

```
    cmdL: out event port ;
    return: out event port ;
    switchL: in event port;
    switchH: in event port;
    cmd: in event port ;
    res: in event port  ;
end s_dispatch;

subject implementation s_dispatch.i
end s_dispatch.i;

subject s_user
  features
    switchL: out event port;
    switchH: out event port;
    cmd: out event port ;
    return: in event port ;
end s_user;

subject implementation s_user.i
end s_user.i;

-------------------------------------------------------------------------------
-- Declaration of nodes
-------------------------------------------------------------------------------
node n_tt1
  features
    rao_cmdH: in event port ;
    rao_cmdL: in event port ;
    rao_res: out event port ;
end n_tt1;

node implementation n_tt1.recons
  subcomponents
    ram_high: memory m_high_ram ;
    disk_high: memory m_high_disk ;
    high: subject s_high accesses ram_high accesses disk_high ;

    ram_low: memory m_low_ram ;
    disk_low: memory m_low_disk ;
    low: subject s_low accesses ram_low accesses disk_low ;

    ram_mns: memory m_mns_ram;
  connections
    port low.resL -> high.resL ;
    port rao_cmdL -> low.cmdL ;
    port rao_cmdH -> high.cmdH ;
    port high.res -> rao_res ;
end n_tt1.recons;


node n_tt2
  features
```

```
    rao_cmdH: out event port ;
    rao_cmdL: out event port ;
    rao_res:  in event port ;
end n_tt2;

node implementation n_tt2.recons
  subcomponents
    ram_dispatch: memory m_dispatch_ram ;
    disk_dispatch: memory m_dispatch_disk ;
    dispatch: subject s_dispatch accesses ram_dispatch accesses disk_dispatch ;

    ram_user: memory m_user_ram ;
    disk_user: memory m_user_disk ;
    user: subject s_user accesses ram_user accesses disk_user ;

    ram_mns: memory m_mns_ram;
  connections
    port user.cmd -> dispatch.cmd ;
    port user.switchH -> dispatch.switchH ;
    port user.switchL -> dispatch.switchL ;
    port dispatch.return -> user.return ;
    port dispatch.cmdH -> rao_cmdH ;
    port dispatch.cmdL -> rao_cmdL ;
    port rao_res -> dispatch.res ;
end n_tt2.recons;


-------------------------------------------------------------------------------
-- Declaration of the main system
-------------------------------------------------------------------------------

system main
end main;

system implementation main.i
  subcomponents
    tt1:  node n_tt1.recons ;
    tt2:  node n_tt2.recons ;
  connections
    port tt2.rao_cmdH -> tt1.rao_cmdH ;
    port tt2.rao_cmdL -> tt1.rao_cmdL ;
    port tt1.rao_res  -> tt2.rao_res  ;
end main.i;
```

C Sample Separation Kernel Configuration

The following is the configuration for the node tt1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    XML Creation Date: 2014.Nov.26-16:45:04

    Command Line:  mkcv -sskt=0001 tt1 -subject-vds0=TTE0,usb*
           -subject-fv1=mem=2G,patabde=wwn-0x5000c5006da0c046-part2,e1000=fv2,rao=rao1
           -subject-fv2=mem=1G,patabde=wwn-0x5000c5006da0c046-part3,e1000=fv1,rao=rao2 -kvm


    Subject "fv1", type FULLVIRT, memory 2.00GiB, 1 vCPU(s)
            initparams=ATA:1 ATA:0 ATA:2 bde_shmem=BDE_SHMEM0 e1000_shmem=NIC_MEM_1 e1000_shmem_1=NIC_MEM_2
    ================================================================================
    Name            Vendor   Device   BDF        IRQ CanSep       Flag Description
    ====            ======   ======   ===        === ======       ==== ===========
    KVMDEV0         0000     0303     00:01.1   1/12     No            Virtual Keyboard/Video/Mouse
    VIRTVGA0        0123     4d08     00:01.2                          Virtual VGA
    E1000_1         8086     100e     00:05.0     16     No            Virtual Ethernet Controller:
            Intel Corporation 82540EM Gigabit Ethernet Controller #1 (fv1/fv2 over NIC_MEM_1, MAC=02-00-02-D0-98-05)
    E1000_2         8086     100e     00:06.0     17     No            Virtual Ethernet Controller:
            Intel Corporation 82540EM Gigabit Ethernet Controller #2 (fv1/vds0 over NIC_MEM_2, MAC=02-00-03-51-AC-06)
    ================================================================================

    Subject "fv2", type FULLVIRT, memory 1.00GiB, 1 vCPU(s)
            initparams=ATA:1 ATA:0 ATA:2 bde_shmem=BDE_SHMEM1 e1000_shmem=NIC_MEM_1 e1000_shmem_1=NIC_MEM_3
    ================================================================================
    Name            Vendor   Device   BDF        IRQ CanSep       Flag Description
    ====            ======   ======   ===        === ======       ==== ===========
    KVMDEV1         0000     0303     00:01.1   1/12     No            Virtual Keyboard/Video/Mouse
    VIRTVGA1        0123     4d08     00:01.2                          Virtual VGA
    E1000_3         8086     100e     00:05.0     16     No            Virtual Ethernet Controller:
Intel Corporation 82540EM Gigabit Ethernet Controller #1 (fv2/fv1 over NIC_MEM_1, MAC=02-00-01-5B-99-05)
    E1000_4         8086     100e     00:06.0     17     No            Virtual Ethernet Controller:
Intel Corporation 82540EM Gigabit Ethernet Controller #2 (fv2/vds0 over NIC_MEM_3, MAC=02-00-00-EB-2B-06)
    ================================================================================

    Subject "vds0", type 64BIT_PARAVIRT, memory 1.00GiB, 1 vCPU(s)
            vmlinux=vds-kernel-amd64.bin
            ramdisk=vds-ramdisk-amd64-complete.img
            initparams=disableapic nomodeset quiet lsk_subject_name=vds0 nodename=tt1 libata.allow_tpm=1
 lsk_ramdisk_region=vds0_RAMDISK lsk_extra_memory=vds0_lowmem acpi=off kvmopts=1-1-80:2-1-80:0-1-19 video=kvmfb:0xE0A00000:640:480
    ================================================================================
    Name            Vendor   Device   BDF        IRQ CanSep       Flag Description
    ====            ======   ======   ===        === ======       ==== ===========
    ETH0            8086     1502     00:19.0     20  PV,FV            Ethernet controller:
                                        Intel Corporation 82579LM Gigabit Network Connection (rev 06)
    USB0            8086     1d2d     00:1a.0     16     FV            USB controller:
                        Intel Corporation X79 series chipset USB2 Enhanced Host Controller #2 (rev 06) (prog-if 20 [EHCI])
    USB1            8086     1d26     00:1d.0     23     FV            USB controller:
                        Intel Corporation X79 series chipset USB2 Enhanced Host Controller #1 (rev 06) (prog-if 20 [EHCI])
    USB1DEV0        413c     2107                                      Dell Computer Corp.  (PID: 00.1d.0-1-1.6)
    SATA0           8086     1d02     00:1f.2     19  PV,FV            SATA controller:
                        Intel Corporation X79 series chipset 6-Port SATA AHCI Controller (rev 06) (prog-if 01 [AHCI 1.0])
    GRAPHICS0       1002     6809     02:00.0     32  PV,FV    Primary VGA compatible controller:
                        Advanced Micro Devices [AMD] nee ATI Device 6809 (prog-if 00 [VGA controller])
    TTE0            1172     675c     03:00.0     40  PV,FV            Unassigned class [ff00]:
                        Altera Corporation Device 675c (rev 02)
    VGA             0000     0900                                      Standard VGA
    ================================================================================

    System Memory Usage:
        RESERVED0 : 156.00KiB
        ibit1     : 1.23MiB
        ibit0     : 1.23MiB
        ibit2     : 1.23MiB
        fv2       : 1.01GiB
        vds0      : 1.39GiB
        fv1       : 2.01GiB
        LSK       : 4.41GiB

-->

<sskt hcvschemaversion="SSKT_HCV_Schema_Version_0001">
  <lsenvironment iommu="true" nodename="tt1" rifpath="rif.bin" skhheapsize="0000000004000000"
    skhpath="lsk.bin" smidisable="false" smt="false" tickspersec="1000">
    <diagoutput vga="true">
      <serial baudrate="115200" lcr="03" maxbaudrate="115200" port="03f8"/>
    </diagoutput>
  </lsenvironment>
  <systemresources>
    <mainmemory hostmemname="host_main_memory">
      <memblock memblocksize="000000000009E000" memblockstart="0000000000000000"/>
      <memblock memblocksize="0000000087DD6000" memblockstart="0000000000100000"/>
      <memblock memblocksize="0000000000775000" memblockstart="0000000088000000"/>
      <memblock memblocksize="00000000006E8000" memblockstart="0000000088800000"/>
      <memblock memblocksize="000000000159F000" memblockstart="0000000089000000"/>
```

```
      <memblock memblocksize="0000000001D93000" memblockstart="000000008A800000"/>
      <memblock memblocksize="0000000000023000" memblockstart="000000008C742000"/>
      <memblock memblocksize="0000000000815000" memblockstart="000000008C7EB000"/>
      <memblock memblocksize="0000000370000000" memblockstart="0000000100000000"/>
    </mainmemory>
    <processor processorname="phys_core_0"/>
    <processor hwid="02" processorname="phys_core_1"/>
    <processor hwid="04" processorname="phys_core_2"/>
  </systemresources>
  <partition pname="unclassified_partition"/>
  <sibitschedpolicy policyname="sibitschedpolicy">
    <majorframe processorname="phys_core_0">
      <minorframe timeslice="4" vpname="ibit0_vcpu_0"/>
    </majorframe>
    <majorframe processorname="phys_core_1">
      <minorframe timeslice="4" vpname="ibit1_vcpu_0"/>
    </majorframe>
    <majorframe processorname="phys_core_2">
      <minorframe timeslice="4" vpname="ibit2_vcpu_0"/>
    </majorframe>
  </sibitschedpolicy>
  <schedulingpolicy policyname="default">
    <majorframe processorname="phys_core_0">
      <minorframe timeslice="30" vpname="vds0_vcpu_0"/>
    </majorframe>
    <majorframe processorname="phys_core_1">
      <minorframe timeslice="30" vpname="fv1_vcpu_0"/>
    </majorframe>
    <majorframe processorname="phys_core_2">
      <minorframe timeslice="30" vpname="fv2_vcpu_0"/>
    </majorframe>
  </schedulingpolicy>
  <exportedresources>
    <platformfeature featurename="PAT" pname="unclassified_partition"/>
    <platformfeature featurename="PCIE" pname="unclassified_partition"/>
    <memoryregion memorytype="RMRR" memregionname="RMRR0" pname="unclassified_partition" size="0000000000027000">
      <starthpa>000000008C6FD000</starthpa>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="vds0_init_image" pname="unclassified_partition" size="0000000040000000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PML4" memregionname="vds0_PML4" pname="unclassified_partition" size="0000000000400000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PDPT" memregionname="vds0_PDPT" pname="unclassified_partition" size="0000000000400000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="vds0_lowmem" pname="unclassified_partition" size="000000000C800000">
      <alignment>2MByteLow</alignment>
    </memoryregion>
    <memoryregion memorytype="PTE" memregionname="vds0_PTE" pname="unclassified_partition" size="0000000004000000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PDT" memregionname="vds0_PDT" pname="unclassified_partition" size="0000000000800000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="BOOTSTRAP" memregionname="BOOTSTRAP" pname="unclassified_partition" size="0000000000001000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="BIOS" memregionname="BIOS" pname="unclassified_partition" size="0000000000020000">
      <starthpa>00000000000E0000</starthpa>
    </memoryregion>
    <memoryregion memorytype="ROPAGE" memregionname="vds0_ROPAGE" pname="unclassified_partition" size="0000000000010000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="ARGPAGE" memregionname="vds0_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="BIOS" memregionname="LOWMEM" pname="unclassified_partition" size="00000000000A0000">
      <starthpa>0000000000000000</starthpa>
    </memoryregion>
    <memoryregion memorytype="BIOS" memregionname="VGABUF" pname="unclassified_partition" size="0000000000020000">
      <starthpa>00000000000A0000</starthpa>
    </memoryregion>
    <memoryregion memorytype="BIOS" memregionname="VIDEO_BIOS" pname="unclassified_partition" size="0000000000020000">
      <starthpa>00000000000C0000</starthpa>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM1" pname="unclassified_partition" size="0000000000200000">
      <alignment>2MByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM2" pname="unclassified_partition" size="00000000000A0000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM3" pname="unclassified_partition" size="0000000000020000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM4" pname="unclassified_partition" size="0000000000100000">
      <alignment>4KByte</alignment>
    </memoryregion>
    <memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM5" pname="unclassified_partition" size="0000000010000000">
      <alignment>2MByte</alignment>
```

```
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM6" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM7" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM8" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM9" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM10" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM11" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv1_fv1:RAM12" pname="unclassified_partition" size="000000000FE00000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PTE" memregionname="fv1_PTE" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PDT" memregionname="fv1_PDT" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PDPT" memregionname="fv1_PDPT" pname="unclassified_partition" size="0000000000008000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PML4" memregionname="fv1_PML4" pname="unclassified_partition" size="0000000000008000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ROPAGE" memregionname="fv1_ROPAGE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ARGPAGE" memregionname="fv1_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="BDE_SHMEM0" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM1" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM2" pname="unclassified_partition" size="00000000000A0000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM3" pname="unclassified_partition" size="0000000000020000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM4" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM5" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM6" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM7" pname="unclassified_partition" size="0000000010000000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="fv2_fv2:RAM8" pname="unclassified_partition" size="000000000FE00000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PTE" memregionname="fv2_PTE" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PDT" memregionname="fv2_PDT" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PDPT" memregionname="fv2_PDPT" pname="unclassified_partition" size="0000000000008000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PML4" memregionname="fv2_PML4" pname="unclassified_partition" size="0000000000008000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ROPAGE" memregionname="fv2_ROPAGE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ARGPAGE" memregionname="fv2_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="BDE_SHMEM1" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="KVM0" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
```

```
<memoryregion memorytype="BIOS" memregionname="VIDEO_FB0" pname="unclassified_partition" size="0000000000500000">
  <starthpa>00000000E0000000</starthpa>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="KVM1" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="BIOS" memregionname="VIDEO_FB1" pname="unclassified_partition" size="0000000000500000">
  <starthpa>00000000E0500000</starthpa>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="NIC_MEM_1" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="NIC_MEM_2" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion fill="00" memorytype="VDEVIO" memregionname="NIC_MEM_3" pname="unclassified_partition" size="0000000000200000">
  <alignment>2MByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="ibit0_init_image" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PTE" memregionname="ibit0_PTE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PDT" memregionname="ibit0_PDT" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="BITRESULTS" memregionname="ibit0_BITRESULTS" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ROPAGE" memregionname="ibit0_ROPAGE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ARGPAGE" memregionname="ibit0_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="ibit1_init_image" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PTE" memregionname="ibit1_PTE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PDT" memregionname="ibit1_PDT" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="BITRESULTS" memregionname="ibit1_BITRESULTS" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ROPAGE" memregionname="ibit1_ROPAGE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ARGPAGE" memregionname="ibit1_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PROGRAM" memregionname="ibit2_init_image" pname="unclassified_partition" size="0000000000100000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PTE" memregionname="ibit2_PTE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="PDT" memregionname="ibit2_PDT" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="BITRESULTS" memregionname="ibit2_BITRESULTS" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ROPAGE" memregionname="ibit2_ROPAGE" pname="unclassified_partition" size="0000000000010000">
  <alignment>4KByte</alignment>
</memoryregion>
<memoryregion memorytype="ARGPAGE" memregionname="ibit2_ARGPAGE" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</memoryregion>
<guestimage memorytype="BOOT" memregionname="vds0_init_image_boot" path="vds-kernel-amd64.bin" pname="unclassified_partition"
                                                                                                  size="0000000000326000">
  <alignment>2MByte</alignment>
</guestimage>
<guestimage memorytype="PROGRAM" memregionname="vds0_RAMDISK" path="vds-ramdisk-amd64-complete.img" pname="unclassified_partition"
                                                                                                  size="00000000068BD000">
  <alignment>2MByte</alignment>
</guestimage>
<guestimage memorytype="BOOT" memregionname="fv1_init_image_boot" path="fvs.bin" pname="unclassified_partition" size="0000000000071000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="BOOT" memregionname="fv2_init_image_boot" path="fvs.bin" pname="unclassified_partition" size="0000000000071000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="BOOT" memregionname="ibit0_init_image_boot" path="ibit.bin" pname="unclassified_partition" size="000000000000A000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="BOOT" memregionname="ibit1_init_image_boot" path="ibit.bin" pname="unclassified_partition" size="000000000000A000">
  <alignment>4KByte</alignment>
```

```
</guestimage>
<guestimage memorytype="BOOT" memregionname="ibit2_init_image_boot" path="ibit.bin" pname="unclassified_partition" size="000000000000A000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="PROGRAM" memregionname="VDSCONFIG" path="tt1-vdsconfig.yml" pname="unclassified_partition" size="0000000000004000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="PROGRAM" memregionname="GIFP" path="tt1.gfp" pname="unclassified_partition" size="0000000000001000">
  <alignment>4KByte</alignment>
</guestimage>
<guestimage memorytype="PROGRAM" memregionname="TTEHEX" path="tt1.tte" pname="unclassified_partition" size="0000000000030000">
  <alignment>4KByte</alignment>
</guestimage>
<externaldevice bridgenum="1" busnum="0" devicenum="22" devid="80861d3a" externaldevname="COMM0" funcnum="0" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="22" devid="80861d3c" externaldevname="IDE0" funcnum="2" pname="unclassified_partition">
  <deviceio barnum="0" devionamesuffix="_DEV_IO0" iostartaddr="F0F0">
    <ioportrange fromoffset="0000" tooffset="0007"/>
  </deviceio>
  <deviceio barnum="1" devionamesuffix="_DEV_IO1" iostartaddr="F0E0">
    <ioportrange fromoffset="0000" tooffset="0003"/>
  </deviceio>
  <deviceio barnum="2" devionamesuffix="_DEV_IO2" iostartaddr="F0D0">
    <ioportrange fromoffset="0000" tooffset="0007"/>
  </deviceio>
  <deviceio barnum="3" devionamesuffix="_DEV_IO3" iostartaddr="F0C0">
    <ioportrange fromoffset="0000" tooffset="0003"/>
  </deviceio>
  <deviceio barnum="4" devionamesuffix="_DEV_IO4" iostartaddr="F0B0">
    <ioportrange fromoffset="0000" tooffset="000F"/>
  </deviceio>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="22" devid="80861d3d" externaldevname="UART0" funcnum="3" pname="unclassified_partition">
  <deviceio barnum="0" devionamesuffix="_DEV_IO0" iostartaddr="F0A0">
    <ioportrange fromoffset="0000" tooffset="0007"/>
  </deviceio>
  <devicememory barnum="1" devmemnamesuffix="_DEV_MEM1"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="25" devid="80861502" externaldevname="ETH0" funcnum="0" pname="unclassified_partition">
  <deviceio barnum="2" devionamesuffix="_DEV_IO2" iostartaddr="F040">
    <ioportrange fromoffset="0000" tooffset="001F"/>
  </deviceio>
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
  <devicememory barnum="1" devmemnamesuffix="_DEV_MEM1"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="26" devid="80861d2d" externaldevname="USB0" funcnum="0" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="27" devid="80861d20" externaldevname="AUDIO0" funcnum="0" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="29" devid="80861d26" externaldevname="USB1" funcnum="0" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="0" devicenum="31" devid="80861d02" externaldevname="SATA0" funcnum="2" pname="unclassified_partition">
  <deviceio barnum="0" devionamesuffix="_DEV_IO0" iostartaddr="F090">
    <ioportrange fromoffset="0000" tooffset="0007"/>
  </deviceio>
  <deviceio barnum="1" devionamesuffix="_DEV_IO1" iostartaddr="F080">
    <ioportrange fromoffset="0000" tooffset="0003"/>
  </deviceio>
  <deviceio barnum="2" devionamesuffix="_DEV_IO2" iostartaddr="F070">
    <ioportrange fromoffset="0000" tooffset="0007"/>
  </deviceio>
  <deviceio barnum="3" devionamesuffix="_DEV_IO3" iostartaddr="F060">
    <ioportrange fromoffset="0000" tooffset="0003"/>
  </deviceio>
  <deviceio barnum="4" devionamesuffix="_DEV_IO4" iostartaddr="F020">
    <ioportrange fromoffset="0000" tooffset="001F"/>
  </deviceio>
  <devicememory barnum="5" devmemnamesuffix="_DEV_MEM5"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="2" devicenum="0" devid="10026809" externaldevname="GRAPHICS0" funcnum="0" pname="unclassified_partition">
  <deviceio barnum="4" devionamesuffix="_DEV_IO4" iostartaddr="E000">
    <ioportrange fromoffset="0000" tooffset="00FF"/>
  </deviceio>
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
  <devicememory barnum="2" devmemnamesuffix="_DEV_MEM2"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="2" devicenum="0" devid="1002aab0" externaldevname="AUDIO1" funcnum="1" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
</externaldevice>
<externaldevice bridgenum="1" busnum="3" devicenum="0" devid="1172675c" externaldevname="TTE0" funcnum="0" pname="unclassified_partition">
  <devicememory barnum="0" devmemnamesuffix="_DEV_MEM0"/>
  <devicememory barnum="1" devmemnamesuffix="_DEV_MEM1"/>
</externaldevice>
<externaldevice bridgenum="0" devid="00000900" externaldevname="VGA" pname="unclassified_partition">
  <deviceio barnum="0" devionamesuffix="_DEV_IO0" iostartaddr="03C0">
    <ioportrange fromoffset="0000" tooffset="001F"/>
  </deviceio>
```

```
    <deviceio barnum="1" devionamesuffix="_DEV_IO1" iostartaddr="03B0">
      <ioportrange fromoffset="0000" tooffset="000B"/>
    </deviceio>
    <deviceio barnum="2" devionamesuffix="_DEV_IO2" iostartaddr="01CE">
      <ioportrange fromoffset="0000" tooffset="0001"/>
    </deviceio>
  </externaldevice>
  <virtualdevice pname="unclassified_partition" vdevname="VBD0">
    <pciaddress busnum="0" devnum="4" funcnum="0"/>
    <bar>
      <ioports baseport="01F0" range="8"/>
    </bar>
    <bar>
      <ioports baseport="03F0" range="8"/>
    </bar>
    <bar>
      <ioports baseport="A800" range="8"/>
    </bar>
    <bar>
      <ioports baseport="A480" range="4"/>
    </bar>
    <bar>
      <ioports baseport="A400" range="16"/>
    </bar>
    <bar>
      <ioports baseport="A080" range="16"/>
    </bar>
    <interface pciclass="01018A" pcicommand="04" pciid="01234d04"/>
    <interface pciclass="01018A" pcicommand="04" pciid="01234d04">
      <capflow>
        <irq>128</irq>
      </capflow>
      <capconfig>
        <server>vds0</server>
      </capconfig>
    </interface>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="VBD1">
    <pciaddress busnum="0" devnum="4" funcnum="0"/>
    <bar>
      <ioports baseport="01F0" range="8"/>
    </bar>
    <bar>
      <ioports baseport="03F0" range="8"/>
    </bar>
    <bar>
      <ioports baseport="A800" range="8"/>
    </bar>
    <bar>
      <ioports baseport="A480" range="4"/>
    </bar>
    <bar>
      <ioports baseport="A400" range="16"/>
    </bar>
    <bar>
      <ioports baseport="A080" range="16"/>
    </bar>
    <interface pciclass="01018A" pcicommand="04" pciid="01234d04"/>
    <interface pciclass="01018A" pcicommand="04" pciid="01234d04">
      <capflow>
        <irq>129</irq>
      </capflow>
      <capconfig>
        <server>vds0</server>
      </capconfig>
    </interface>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="KVMDEV0">
    <pciaddress busnum="0" devnum="1" funcnum="1"/>
    <interface irq="0" pciid="00000000">
      <emulatelegacy devid="00000303" irq="1">
        <ioaccessmethod>
          <ioports baseport="0060" range="1"/>
        </ioaccessmethod>
        <ioaccessmethod>
          <ioports baseport="0064" range="1"/>
        </ioaccessmethod>
      </emulatelegacy>
      <emulatelegacy devid="00000F13" irq="12"/>
    </interface>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="VIRTVGA0">
    <pciaddress busnum="0" devnum="1" funcnum="2"/>
    <bar>
      <iomem memaddr="0000000080000000" size="00800000"/>
    </bar>
    <interface pciclass="030000" pcicommand="23" pciid="01234d08"/>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="KVMDEV1">
    <pciaddress busnum="0" devnum="1" funcnum="1"/>
    <interface irq="0" pciid="00000000">
```

```
      <emulatelegacy devid="00000303" irq="1">
        <ioaccessmethod>
          <ioports baseport="0060" range="1"/>
        </ioaccessmethod>
        <ioaccessmethod>
          <ioports baseport="0064" range="1"/>
        </ioaccessmethod>
      </emulatelegacy>
      <emulatelegacy devid="00000F13" irq="12"/>
    </interface>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="VIRTVGA1">
    <pciaddress busnum="0" devnum="1" funcnum="2"/>
    <bar>
      <iomem memaddr="0000000040000000" size="00800000"/>
    </bar>
    <interface pciclass="030000" pcicommand="23" pciid="01234d08"/>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="E1000_1">
    <pciaddress busnum="0" devnum="5" funcnum="0"/>
    <bar>
      <iomem memaddr="0000000081100000" size="00020000"/>
    </bar>
    <bar>
      <ioports baseport="7000" range="8"/>
    </bar>
    <interface irq="16" pciclass="020000" pcicommand="07" pciid="8086100e">
      <capflow>
        <irq>130</irq>
      </capflow>
      <capconfig>
        <server>fv2</server>
      </capconfig>
      <capconfig>
        <macaddr>02-00-02-D0-98-05</macaddr>
      </capconfig>
    </interface>
    <interface pciclass="020000" pciid="8086100e"/>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="E1000_3">
    <pciaddress busnum="0" devnum="5" funcnum="0"/>
    <bar>
      <iomem memaddr="0000000041100000" size="00020000"/>
    </bar>
    <bar>
      <ioports baseport="7000" range="8"/>
    </bar>
    <interface irq="16" pciclass="020000" pcicommand="07" pciid="8086100e">
      <capflow>
        <irq>130</irq>
      </capflow>
      <capconfig>
        <server>fv1</server>
      </capconfig>
      <capconfig>
        <macaddr>02-00-01-5B-99-05</macaddr>
      </capconfig>
    </interface>
    <interface pciclass="020000" pciid="8086100e"/>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="E1000_2">
    <pciaddress busnum="0" devnum="6" funcnum="0"/>
    <bar>
      <iomem memaddr="0000000081120000" size="00020000"/>
    </bar>
    <bar>
      <ioports baseport="7040" range="8"/>
    </bar>
    <interface irq="17" pciclass="020000" pcicommand="07" pciid="8086100e">
      <capflow>
        <irq>131</irq>
      </capflow>
      <capconfig>
        <server>vds0</server>
      </capconfig>
      <capconfig>
        <macaddr>02-00-03-51-AC-06</macaddr>
      </capconfig>
      <capconfig>
        <rao>rao1</rao>
      </capconfig>
      <capconfig>
        <nodename>tt1</nodename>
      </capconfig>
    </interface>
    <interface pciclass="020000" pciid="8086100e"/>
  </virtualdevice>
  <virtualdevice pname="unclassified_partition" vdevname="E1000_4">
    <pciaddress busnum="0" devnum="6" funcnum="0"/>
    <bar>
      <iomem memaddr="0000000041120000" size="00020000"/>
```

```
      </bar>
      <bar>
        <ioports baseport="7040" range="8"/>
      </bar>
      <interface irq="17" pciclass="020000" pcicommand="07" pciid="8086100e">
        <capflow>
          <irq>132</irq>
        </capflow>
        <capconfig>
          <server>vds0</server>
        </capconfig>
        <capconfig>
          <macaddr>02-00-00-EB-2B-06</macaddr>
        </capconfig>
        <capconfig>
          <rao>rao2</rao>
        </capconfig>
        <capconfig>
          <nodename>tt1</nodename>
        </capconfig>
      </interface>
      <interface pciclass="020000" pciid="8086100e"/>
    </virtualdevice>
    <messagebuffer buffersize="64" msgbufname="ibit0_to_ibit1" pname="unclassified_partition"/>
    <messagebuffer buffersize="64" msgbufname="ibit1_to_ibit0" pname="unclassified_partition"/>
    <messagebuffer buffersize="64" msgbufname="ibit0_to_ibit2" pname="unclassified_partition"/>
    <messagebuffer buffersize="64" msgbufname="ibit2_to_ibit0" pname="unclassified_partition"/>
    <absoluteclock absclockname="Master_timepiece" pname="unclassified_partition"/>
    <audit auditname="audit_master" maxentries="2048" pname="unclassified_partition">
      <fullbufferaction action="DROP" parameter="write_buffer_to_disk"/>
    </audit>
    <subject initparams="disableapic␣nomodeset␣quiet␣lsk_subject_name=vds0␣nodename=tt1␣libata.allow_tpm=1␣lsk_ramdisk_region=vds0_RAMDISK
␣␣␣␣␣␣␣␣lsk_extra_memory=vds0_lowmem␣acpi=off␣kvmopts=1-1-80:2-1-80:0-1-19␣video=kvmfb:0xE0A00000:640:480" irqbase="48" mode="64BIT_PARAVIRT"
      pname="unclassified_partition" readschedpolicyperm="ALLOW" sname="vds0" startaddr="FFFFFFFF80000000" writeschedpolicyperm="ALLOW">
      <memoryflow guestaddr="FFFFFFFF80000000" guestaddrsize="02800000" memregname="vds0_init_image" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="FFFFFFFF01000000" memregname="vds0_init_image_boot" readperm="ALLOW"/>
      <memoryflow memregname="vds0_RAMDISK" readperm="ALLOW"/>
      <memoryflow memregname="vds0_PML4" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="vds0_PDPT" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="vds0_lowmem" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="vds0_PTE" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="vds0_PDT" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="FFFFFFFF02000000" memregname="BOOTSTRAP" readperm="ALLOW"/>
      <memoryflow guestaddr="00000000003D0000" memregname="BIOS" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="FFFFFFFF00000000" memregname="vds0_ROPAGE" readperm="ALLOW"/>
      <memoryflow guestaddr="FFFFFFFF00100000" memregname="vds0_ARGPAGE" readperm="ALLOW"/>
      <memoryflow guestaddr="0000000000300000" memregname="LOWMEM" readperm="ALLOW"/>
      <memoryflow memregname="VGABUF" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="00000000003F0000" memregname="VIDEO_BIOS" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="RMRR0" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000100000" memregname="BDE_SHMEM0" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000200000" memregname="BDE_SHMEM1" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="KVM0" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="KVM1" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="NIC_MEM_2" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="NIC_MEM_3" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000410000" memregname="VDSCONFIG" readperm="ALLOW"/>
      <memoryflow guestaddr="0000000000414000" memregname="GIFP" readperm="ALLOW"/>
      <memoryflow guestaddr="00000000003A0000" memregname="TTEHEX" readperm="ALLOW"/>
      <externaldeviceflow externaldevname="TTE0" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="USB0" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="USB1" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="SATA0" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="GRAPHICS0" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="VGA" readperm="ALLOW" writeperm="ALLOW"/>
      <externaldeviceflow externaldevname="ETH0" readperm="ALLOW" writeperm="ALLOW"/>
      <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" writeabsclockperm="DENY"/>
      <subjectflow restartperm="ALLOW" resumeperm="ALLOW" sname="fv1" startperm="ALLOW" statusperm="ALLOW" stopperm="ALLOW" suspendperm="ALLOW">
        <injectintperm irq="128" perm="ALLOW"/>
        <injectintperm irq="1" perm="ALLOW"/>
        <injectintperm irq="12" perm="ALLOW"/>
        <injectintperm irq="131" perm="ALLOW"/>
      </subjectflow>
      <subjectflow restartperm="ALLOW" resumeperm="ALLOW" sname="fv2" startperm="ALLOW" statusperm="ALLOW" stopperm="ALLOW" suspendperm="ALLOW">
        <injectintperm irq="129" perm="ALLOW"/>
        <injectintperm irq="1" perm="ALLOW"/>
        <injectintperm irq="12" perm="ALLOW"/>
        <injectintperm irq="132" perm="ALLOW"/>
      </subjectflow>
      <virtualprocessor vpname="vds0_vcpu_0"/>
    </subject>
    <subject initparams="ATA:1␣ATA:0␣ATA:2␣bde_shmem=BDE_SHMEM0␣e1000_shmem=NIC_MEM_1␣e1000_shmem_1=NIC_MEM_2" mode="FULLVIRT"
      pname="unclassified_partition" sname="fv1" startaddr="FFE00000">
      <platformfeatureflow featurename="PAT" readperm="ALLOW" writeperm="ALLOW"/>
      <platformfeatureflow featurename="PCIE" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="00000000FFE00000" memregname="fv1_fv1:RAM1" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000000000" memregname="fv1_fv1:RAM2" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="00000000000E0000" memregname="fv1_fv1:RAM3" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000100000" memregname="fv1_fv1:RAM4" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000000200000" memregname="fv1_fv1:RAM5" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="0000000010200000" memregname="fv1_fv1:RAM6" readperm="ALLOW" writeperm="ALLOW"/>
```

```
    <memoryflow guestaddr="0000000020200000" memregname="fv1_fv1:RAM7" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000030200000" memregname="fv1_fv1:RAM8" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000040200000" memregname="fv1_fv1:RAM9" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000050200000" memregname="fv1_fv1:RAM10" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000060200000" memregname="fv1_fv1:RAM11" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000070200000" memregname="fv1_fv1:RAM12" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow memregname="fv1_init_image_boot" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000080800000" memregname="fv1_PTE" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000080A00000" memregname="fv1_PDT" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000081150000" memregname="fv1_PDPT" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000081158000" memregname="fv1_PML4" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000081140000" memregname="fv1_ROPAGE" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000081160000" memregname="fv1_ARGPAGE" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000081000000" memregname="BDE_SHMEM0" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000081161000" memregname="KVM0" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow caching="WRITE-COMBINING" guestaddr="0000000080000000" memregname="VIDEO_FB0" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000080C00000" memregname="NIC_MEM_1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000080E00000" memregname="NIC_MEM_2" readperm="ALLOW" writeperm="ALLOW"/>
    <virtualdeviceflow interface="0" vdevname="KVMDEV0"/>
    <virtualdeviceflow interface="0" vdevname="VIRTVGA0"/>
    <virtualdeviceflow interface="0" vdevname="E1000_1"/>
    <virtualdeviceflow interface="0" vdevname="E1000_2"/>
    <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" writeabsclockperm="DENY"/>
    <subjectflow sname="vds0">
      <injectintperm irq="128" perm="ALLOW"/>
      <injectintperm irq="1" perm="ALLOW"/>
      <injectintperm irq="12" perm="ALLOW"/>
      <injectintperm irq="131" perm="ALLOW"/>
    </subjectflow>
    <subjectflow sname="fv2">
      <injectintperm irq="130" perm="ALLOW"/>
    </subjectflow>
    <virtualprocessor vpname="fv1_vcpu_0"/>
  </subject>
  <subject initparams="ATA:1␣ATA:0␣ATA:2␣bde_shmem=BDE_SHMEM1␣e1000_shmem=NIC_MEM_1␣e1000_shmem_1=NIC_MEM_3" mode="FULLVIRT"
    pname="unclassified_partition" sname="fv2" startaddr="FFE00000">
    <platformfeatureflow featurename="PAT" readperm="ALLOW" writeperm="ALLOW"/>
    <platformfeatureflow featurename="PCIE" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="00000000FFE00000" memregname="fv2_fv2:RAM1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000000000000" memregname="fv2_fv2:RAM2" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="00000000000E0000" memregname="fv2_fv2:RAM3" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000000100000" memregname="fv2_fv2:RAM4" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000000200000" memregname="fv2_fv2:RAM5" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000010200000" memregname="fv2_fv2:RAM6" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000020200000" memregname="fv2_fv2:RAM7" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000030200000" memregname="fv2_fv2:RAM8" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow memregname="fv2_init_image_boot" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000040800000" memregname="fv2_PTE" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000040A00000" memregname="fv2_PDT" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000041150000" memregname="fv2_PDPT" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000041158000" memregname="fv2_PML4" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000041140000" memregname="fv2_ROPAGE" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000041160000" memregname="fv2_ARGPAGE" readperm="ALLOW"/>
    <memoryflow guestaddr="0000000041000000" memregname="BDE_SHMEM1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="0000000041161000" memregname="KVM1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow caching="WRITE-COMBINING" guestaddr="0000000040000000" memregname="VIDEO_FB1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000040C00000" memregname="NIC_MEM_1" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="000000040E00000" memregname="NIC_MEM_3" readperm="ALLOW" writeperm="ALLOW"/>
    <virtualdeviceflow interface="0" vdevname="KVMDEV1"/>
    <virtualdeviceflow interface="0" vdevname="VIRTVGA1"/>
    <virtualdeviceflow interface="0" vdevname="E1000_3"/>
    <virtualdeviceflow interface="0" vdevname="E1000_4"/>
    <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" writeabsclockperm="DENY"/>
    <subjectflow sname="vds0">
      <injectintperm irq="129" perm="ALLOW"/>
      <injectintperm irq="1" perm="ALLOW"/>
      <injectintperm irq="12" perm="ALLOW"/>
      <injectintperm irq="132" perm="ALLOW"/>
    </subjectflow>
    <subjectflow sname="fv1">
      <injectintperm irq="130" perm="ALLOW"/>
    </subjectflow>
    <virtualprocessor vpname="fv2_vcpu_0"/>
  </subject>
  <subject initparams="serialport=0x3f8␣vga=false␣vgacolor=0x0b␣verbosity=false" mode="SIBIT_MASTER" on_triplefault="maintenance"
    on_watchdog="maintenance" pname="unclassified_partition" sname="ibit0" startaddr="C0000000" watchdog_timeout="1000">
    <memoryflow guestaddr="C0000000" memregname="ibit0_init_image" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow memregname="ibit0_init_image_boot" readperm="ALLOW"/>
    <memoryflow guestaddr="D0800000" memregname="ibit0_PTE" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="D0000000" memregname="ibit0_PDT" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="F0001000" memregname="ibit0_BITRESULTS" readperm="ALLOW" writeperm="ALLOW"/>
    <memoryflow guestaddr="F0002000" memregname="ibit0_ROPAGE" readperm="ALLOW"/>
    <memoryflow guestaddr="F0000000" memregname="ibit0_ARGPAGE" readperm="ALLOW"/>
    <messagebufferflow msgbufname="ibit0_to_ibit1" writeperm="ALLOW"/>
    <messagebufferflow irq="133" msgbufname="ibit1_to_ibit0" readperm="ALLOW"/>
    <messagebufferflow msgbufname="ibit0_to_ibit2" writeperm="ALLOW"/>
    <messagebufferflow irq="135" msgbufname="ibit2_to_ibit0" readperm="ALLOW"/>
    <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" readcalibrationperm="ALLOW" writeabsclockperm="DENY"
      writecalibrationperm="DENY"/>
    <virtualprocessor vpname="ibit0_vcpu_0"/>
  </subject>
```

```
    <subject initparams="serialport=0x3f8␣vga=false␣vgacolor=0x0b␣verbosity=false" mode="SIBIT_SLAVE" on_triplefault="maintenance"
      on_watchdog="maintenance" pname="unclassified_partition" sname="ibit1" startaddr="C0000000" watchdog_timeout="1000">
      <memoryflow guestaddr="C0000000" memregname="ibit1_init_image" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="ibit1_init_image_boot" readperm="ALLOW"/>
      <memoryflow guestaddr="D0800000" memregname="ibit1_PTE" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="D0000000" memregname="ibit1_PDT" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="F0001000" memregname="ibit1_BITRESULTS" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="F0002000" memregname="ibit1_ROPAGE" readperm="ALLOW"/>
      <memoryflow guestaddr="F0000000" memregname="ibit1_ARGPAGE" readperm="ALLOW"/>
      <messagebufferflow irq="134" msgbufname="ibit0_to_ibit1" readperm="ALLOW"/>
      <messagebufferflow msgbufname="ibit1_to_ibit0" writeperm="ALLOW"/>
      <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" readcalibrationperm="ALLOW" writeabsclockperm="DENY"
        writecalibrationperm="DENY"/>
      <virtualprocessor vpname="ibit1_vcpu_0"/>
    </subject>
    <subject initparams="serialport=0x3f8␣vga=false␣vgacolor=0x0b␣verbosity=false" mode="SIBIT_SLAVE" on_triplefault="maintenance"
      on_watchdog="maintenance" pname="unclassified_partition" sname="ibit2" startaddr="C0000000" watchdog_timeout="1000">
      <memoryflow guestaddr="C0000000" memregname="ibit2_init_image" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow memregname="ibit2_init_image_boot" readperm="ALLOW"/>
      <memoryflow guestaddr="D0800000" memregname="ibit2_PTE" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="D0000000" memregname="ibit2_PDT" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="F0001000" memregname="ibit2_BITRESULTS" readperm="ALLOW" writeperm="ALLOW"/>
      <memoryflow guestaddr="F0002000" memregname="ibit2_ROPAGE" readperm="ALLOW"/>
      <memoryflow guestaddr="F0000000" memregname="ibit2_ARGPAGE" readperm="ALLOW"/>
      <messagebufferflow irq="136" msgbufname="ibit0_to_ibit2" readperm="ALLOW"/>
      <messagebufferflow msgbufname="ibit2_to_ibit0" writeperm="ALLOW"/>
      <absoluteclockflow absclockname="Master_timepiece" readabsclockperm="ALLOW" readcalibrationperm="ALLOW" writeabsclockperm="DENY"
        writecalibrationperm="DENY"/>
      <virtualprocessor vpname="ibit2_vcpu_0"/>
    </subject>
  </exportedresources>
</sskt>
```

D Sample MILS Network System Conguration

```
<?xml version="1.0" encoding="UTF-8"?>
<dmils dmilsname="GlobalPolicy1">
<node name="tt1">
    <rao name="rao1" id1="1" id2="0201">
    </rao>
    <rao name="rao2" id1="3" id2="0202">
    </rao>
</node>
<node name="tt2">
    <rao name="rao1" id1="2" id2="a678">
    </rao>
    <rao name="rao2" id1="4" id2="a678">
    </rao>
</node>
<gpf globalflowpolicyname="globalpolicy1">
    <bind fromnode="tt1" fromrao="rao1" tonode="tt2" torao="rao1">
    </bind>
    <bind fromnode="tt1" fromrao="rao2" tonode="tt2" torao="rao2">
    </bind>
</gpf>
</dmils>
```

E Sample Time Triggered Ethernet Network Conguration

```
<?xml version="1.0" encoding="UTF-8"?>
<nd:NetworkDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    interfaceVersionNumber="1"
    transmissionSpeed="1000Mbps"
    redundancy="1"
    ctMarker="AB:AD:BA:BE"
    enableDynamicRouting="true"
    createUnknownDefaultRoutes="true">
  <processingInstructions>
    <schedOption
        Key="seed"
        Value="12345"/>
    <schedOption
        Key="minimum_delta_r"
        Value="2"/>
    <schedOption
        Key="raster_granularity_ns"
        Value="20000"/>
  </processingInstructions>
  <syncDomain
      name="sync_domain"
      clusterPeriod="#//@period[name='PERIOD_CLUSTER']"
      integrationCycleDuration="10000000␣ns"
      faultTolerance="0FTSI_2SM"
      precision="9000␣ns"
      fullCBG="true"
      value="0">
    <syncPriority
        name="SYPRIO_1"
        value="1"/>
  </syncDomain>
  <device
      xsi:type="topo:EndSystem"
      name="ES1"
      syncRole="syncMaster"
      syncPriority="#//@syncDomain/@syncPriority[name='SYPRIO_1']"
      deviceTargets=
      "platform:/plugin/com.tttech.tte.targetdevices/data/ND_Target_Devices.targets#//@targetDeviceDescriptors[name='TTE_PMC_ESys_1G']">
    <port
        name="PORT_ES1_1"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="P1"/>
    <port
        name="PORT_ES1_HOST"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="PHOST"/>
    <port
        name="PORT_ES1_SYNC"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="PSYNC"/>
    <partition
        name="partition_1"
        physicalPort="#//@device[name='ES1']/@port[name='PORT_ES1_HOST']">
      <dataPort
          xsi:type="logical:RxComMacPort"
          name="vl1_rec"
          maxPayloadSize="1500"
          macType="0"/>
      <dataPort
          xsi:type="logical:RxComMacPort"
          name="vl3_rec"
          maxPayloadSize="1500"
          macType="0"/>
      <dataPort
          xsi:type="logical:TxComMacPort"
          name="vl2_snd"
          maxPayloadSize="1500"
          macType="0"/>
      <dataPort
          xsi:type="logical:TxComMacPort"
          name="vl4_snd"
          maxPayloadSize="1500"
          macType="0"/>
    </partition>
  </device>
  <device
      xsi:type="topo:EndSystem"
      name="ES2"
      syncRole="syncMaster"
      syncPriority="#//@syncDomain/@syncPriority[name='SYPRIO_1']"
      deviceTargets=
      "platform:/plugin/com.tttech.tte.targetdevices/data/ND_Target_Devices.targets#//@targetDeviceDescriptors[name='TTE_PMC_ESys_1G']">
    <port
        name="PORT_ES2_1"
```

```
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P1"/>
  <port
      name="PORT_ES2_HOST"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="PHOST"/>
  <port
      name="PORT_ES2_SYNC"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="PSYNC"/>
  <partition
      name="partition_2"
      physicalPort="#//@device[name='ES2']/@port[name='PORT_ES2_HOST']">
    <dataPort
        xsi:type="logical:TxComMacPort"
        name="vl1_snd"
        maxPayloadSize="1500"
        portId=""
        macType="0"/>
    <dataPort
        xsi:type="logical:TxComMacPort"
        name="vl3_snd"
        maxPayloadSize="1500"
        macType="0"/>
    <dataPort
        xsi:type="logical:RxComMacPort"
        name="vl2_rec"
        maxPayloadSize="1500"
        macType="0"/>
    <dataPort
        xsi:type="logical:RxComMacPort"
        name="vl4_rec"
        maxPayloadSize="1500"
        macType="0"/>
  </partition>
</device>
<device
    xsi:type="topo:Switch"
    name="SW0"
    syncRole="syncCompressionMaster"
    syncPriority="#//@syncDomain/@syncPriority[name='SYPRIO_1']"
    deviceTargets=
    "platform:/plugin/com.tttech.tte.targetdevices/data/ND_Target_Devices.targets#//@targetDeviceDescriptors[name='TTE_Dev_Switch_12port_1G']">
  <port
      name="PORT_SW0_1"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P1"/>
  <port
      name="PORT_SW0_2"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P2"/>
  <port
      name="PORT_SW0_3"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P3"/>
  <port
      name="PORT_SW0_4"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P4"/>
  <port
      name="PORT_SW0_5"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P5"/>
  <port
      name="PORT_SW0_6"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P6"/>
  <port
      name="PORT_SW0_7"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P7"/>
  <port
      name="PORT_SW0_8"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P8"/>
  <port
      name="PORT_SW0_9"
      rxLatencyCorrection="0 ns"
      txLatencyCorrection="0 ns"
      type="P9"/>
```

```
    <port
        name="PORT_SW0_10"
        rxLatencyCorrection="0 ns"
        txLatencyCorrection="0 ns"
        type="P10"/>
    <port
        name="PORT_SW0_11"
        rxLatencyCorrection="0 ns"
        txLatencyCorrection="0 ns"
        type="P11"/>
    <port
        name="PORT_SW0_12"
        rxLatencyCorrection="0 ns"
        txLatencyCorrection="0 ns"
        type="P12"/>
    <port
        name="PORT_SW0_SYNC"
        rxLatencyCorrection="0 ns"
        txLatencyCorrection="0 ns"
        type="PSYNC"/>
    <port
        name="PORT_SW0_MGMT"
        rxLatencyCorrection="0 ns"
        txLatencyCorrection="0 ns"
        type="PMGMT"/>
    <bestEffortRoute
        destinationMacAddress="02:02:02:02:00:24"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_MGMT']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="02:02:02:02:08:2F"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_MGMT']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="02:02:02:02:04:2F"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_MGMT']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="00:24:7E:DC:40:19"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_9']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_10']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_11']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="ff:ff:ff:ff:ff:ff"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_9']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="ff:ff:ff:ff:ff:ff"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_10']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="ff:ff:ff:ff:ff:ff"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_11']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="F8:B1:56:C0:00:43"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_9']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="F8:B1:56:B9:14:23"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_10']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="F8:B1:56:C0:87:7E"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_11']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="00:00:00:00:00:00"
        addrMask="00:00:00:00:00:00"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
```

```
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_9']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_10']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_11']"
        channel="1"/>
    <bestEffortRoute
        destinationMacAddress="ff:ff:ff:ff:ff:ff"
        addrMask="ff:ff:ff:ff:ff:ff"
        srcPorts="#//@device[name='SW0']/@port[name='PORT_SW0_12']"
        dstPorts="#//@device[name='SW0']/@port[name='PORT_SW0_9']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_10']
␣␣␣␣␣␣␣␣#//@device[name='SW0']/@port[name='PORT_SW0_11']"
        channel="1"/>
    <managementInterface
        sourceAddress="02:02:02:02:04:2F"
        unlockDestAddress="02:02:02:02:08:2F"
        channel="1">
      <macAcceptanceEntry
         acceptanceMacAddress="02:02:02:02:00:24"
         addressType="nonCriticalTraffic"
         unlockEnabled="true"
         resetEnabled="true"
         responseDestMacAddress="02:02:02:02:08:2F">
        <accessControl
            page="0"
            writeEnable="true"/>
        <accessControl
            page="1"
            writeEnable="true"/>
        <accessControl
            page="2"
            writeEnable="true"/>
        <accessControl
            page="3"
            writeEnable="true"/>
      </macAcceptanceEntry>
    </managementInterface>
  </device>
  <device
      xsi:type="topo:EndSystem"
      name="ES3"
      syncRole="syncMaster"
      syncPriority="#//@syncDomain/@syncPriority[name='SYPRIO_1']"
      deviceTargets=
      "platform:/plugin/com.tttech.tte.targetdevices/data/ND_Target_Devices.targets#//@targetDeviceDescriptors[name='TTE_PMC_ESys_1G']">
    <port
        name="PORT_ES3_1"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="P1"/>
    <port
        name="PORT_ES3_HOST"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="PHOST"/>
    <port
        name="PORT_ES3_SYNC"
        rxLatencyCorrection="0␣ns"
        txLatencyCorrection="0␣ns"
        type="PSYNC"/>
    <partition
        name="partition_3"
        physicalPort="#//@device[name='ES3']/@port[name='PORT_ES3_HOST']"/>
  </device>
  <physicalLink
      name="connection_1"
      transmissionSpeed="1000Mbps"
      mediaType="default"
      port="#//@device[name='ES1']/@port[name='PORT_ES1_1']␣#//@device[name='SW0']/@port[name='PORT_SW0_1']"
      cableLength="2␣m"/>
  <physicalLink
      name="connection_2"
      transmissionSpeed="1000Mbps"
      mediaType="default"
      port="#//@device[name='ES2']/@port[name='PORT_ES2_1']␣#//@device[name='SW0']/@port[name='PORT_SW0_2']"
      cableLength="2␣m"/>
  <physicalLink
      name="connection_3"
      transmissionSpeed="1000Mbps"
      mediaType="default"
      port="#//@device[name='ES3']/@port[name='PORT_ES3_1']␣#//@device[name='SW0']/@port[name='PORT_SW0_3']"
      cableLength="2␣m"/>
  <period
      name="PERIOD_RC_0"
      time="0␣ns"/>
  <period
      name="PERIOD_CLUSTER"
      time="150000000␣ns"/>
  <ethernetLink
      xsi:type="vl:RCVirtualLink"
      name="vl1"
      receivers="#//@device[name='ES1']/@partition[name='partition_1']/@dataPort[name='vl1_rec']"
```

```
      senders="#//@device[name='ES2']/@partition[name='partition_2']/@dataPort[name='vl1_snd']"
      maxFrameSize="1518"
      vlid="1"
      redundancyMgmt="no_check"
      bag="#//@period[name='PERIOD_RC_0']"
      jitter="500 us"
      priority="LOW"/>
  <ethernetLink
      xsi:type="vl:RCVirtualLink"
      name="vl2"
      receivers="#//@device[name='ES2']/@partition[name='partition_2']/@dataPort[name='vl2_rec']"
      senders="#//@device[name='ES1']/@partition[name='partition_1']/@dataPort[name='vl2_snd']"
      maxFrameSize="1518"
      vlid="2"
      redundancyMgmt="no_check"
      bag="#//@period[name='PERIOD_RC_0']"
      jitter="500 us"
      priority="LOW"/>
  <ethernetLink
      xsi:type="vl:RCVirtualLink"
      name="vl3"
      receivers="#//@device[name='ES1']/@partition[name='partition_1']/@dataPort[name='vl3_rec']"
      senders="#//@device[name='ES2']/@partition[name='partition_2']/@dataPort[name='vl3_snd']"
      maxFrameSize="1518"
      vlid="3"
      redundancyMgmt="no_check"
      bag="#//@period[name='PERIOD_RC_0']"
      jitter="500 us"
      priority="LOW"/>
  <ethernetLink
      xsi:type="vl:RCVirtualLink"
      name="vl4"
      receivers="#//@device[name='ES2']/@partition[name='partition_2']/@dataPort[name='vl4_rec']"
      senders="#//@device[name='ES1']/@partition[name='partition_1']/@dataPort[name='vl4_snd']"
      maxFrameSize="1518"
      vlid="4"
      redundancyMgmt="no_check"
      bag="#//@period[name='PERIOD_RC_0']"
      jitter="500 us"
      priority="LOW"/>
</nd:NetworkDescription>
```