Documentation

ASSIST User Guide

Robert Hilbrich, Michael Behrisch

DLR German Aerospace Center
Institute of Transportation Systems

Berlin

DLR
Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

DLR German Aerospace Center
Institute of Transportation Systems
Prof. Dr. Karsten Lemmer
Rutherfordstraße 2
12489 Berlin
Tel: +49 531 295-3401

Robert Hilbrich
Tel: +49 30 67055-582
Fax: +49 30 67055-291
E-Mail: robert.hilbrich@dlr.de

Document Identification:

Title	ASSIST User Guide
Subject	Documentation
Author(s)	Robert Hilbrich, Michael Behrisch
Filename	user-guide.tex
Last saved on	19th September 2015

Document History:

Version 0.1	Initial Version	18.08.2015

Contents

1. Introduction

2. Usage

3. Constraint Semantics

3.1. Basic Elements: RDCs, Connectors, Pins and Equipment Interfaces

- Hardware elements with *containment* relationships:
  - Compartments
  - Remote Data Concentrators - RDCs
  - Connectors
  - Pins
  - Equipment interfaces
- Allocation is generated between *equipment interfaces* and *pins*
- Allocation is also indirectly generated between *equipment interfaces* and all other elements

3.2. Basic Interface Types Constraint

3.2.1. Intended semantics

3.2.2. Input Specification

3.2.3. Implementation Remarks

3.3. Configurable Interface Types Constraint

3.3.1. Intended semantics

3.3.2. Input Specification

3.3.3. Implementation Remarks

3.4. Protection Level Constrains

3.4.1. Intended semantics

3.4.2. Input Specification

3.4.3. Implementation Remarks

3.5. Connected Pins

3.5.1. Intended semantics

3.5.2. Input Specification

3.5.3. Implementation Remarks

A. Mapping Example

In Listing A.1 an example specification is described. This example illustrates the specifications that are possible with the current version of ASSIST.

```
/*
 * GLOBAL PROPERTIES
 */

Global {
    Name = "System";

    Compatible Interface Types {
        "EquipmentType0" −> "PinType0", "PinType1";
        "EquipmentType1" −> "PinType1", "PinType2";
    }

    Cable Weights {
        "EquipmentType0"  = 0.3232;
        "EquipmentType1"  = 1.3232;
        default           = 2.43;
    }

    Protection Level Restrictions {
        RDC.Location = "LocationA" And Equipment.EmhZone1 = "LocationB" −> L1, L3;
        RDC.Location = "LocationB" And Equipment.EmhZone1 = "LocationC" −> L5;
    }
}

/*
 * COMPARTMENTS, RDCs, CONNECTORS and PINS
 */

Compartments {

    Compartment Comp1 {
        Manufacturer    = "ManufacturerName";
        PowerSupply     = "PowerSupplyName";
        Side            = "SideName";
        Zone            = "ZoneName";
```

```
RDC RDC1 {
    Manufacturer    = "ManufacturerName";
    PowerSupply1    = "PowerSupply1Name";
    PowerSupply2    = "PowerSupply2Name";
    Type            = "RDCTypeName";
    ESS             = "ESSName";
    Location        = "RDCLocatioName";
    ResourceX       = −120;
    ResourceY       = 150;
    ResourceZ       = −1200;

    Connector Connector1{
        // Available protection levels:
        // None, L1, L2, L3, L4, L5, L6, L7, L8

        Pin1: "PinType0"; // = protection level None
        Pin2: "PinType0" protection level L5;
        Pin3: "PinType1" protection level L8;

    }

    Connector Connector2{
        Pin1: "PinType3";
        Pin2: "PinType1" protection level L8;
        Pin3: "PinType1" protection level L8;
    }

    // ... more Connectors possible ...

    Connected Pins {
        Comp1.RDC1.Connector1.Pin1, Comp1.RDC1.Connector1.Pin2 are connected;
        Comp1.RDC1.Connector1.Pin3, Comp1.RDC1.Connector2.Pin2, Comp1.RDC1.Connector2
            .Pin3 are connected;
    }

    Metric Parameters {
        "RDC1Parameter1" = 54;

        // ...
    }
}

// ... more RDCs possible ...

Metric Parameters {
    "Comp1Parameter1" = 12;
    "Comp1Parameter2" = 6;
    // ...
}
```

```
    }
}

/*
 * EQUIPMENT INTERFACES
 */

Interfaces {

    Interface EquipmentInterface1 {
        System     = "SystemName";
        SubAta     = "SubAtaName";
        Resource   = "ResourceName";
        LineName   = "LineName";
        WiringLane = "WiringLaneName";
        GrpInfo    = "GrpInfoName";
        Route      = "RouteName";
        PwSup1     = "PwSup1Name";
        EmhZone1   = "EmhZone1Name";
        Type       = "EquipmentType0";
        ResourceX  = −212;
        ResourceY  = −12;
        ResourceZ  = 55;
    }

    Interface EquipmentInterface2 {
        System     = "SystemName";
        SubAta     = "SubAtaName";
        Resource   = "ResourceName";
        LineName   = "LineName";
        WiringLane = "WiringLaneName";
        GrpInfo    = "GrpInfoName";
        Route      = "RouteName";
        PwSup1     = "PwSup1Name";
        EmhZone1   = "EmhZone1Name";
        Type       = "EquipmentType1";
        ResourceX  = −212;
        ResourceY  = −12;
        ResourceZ  = 55;
    }
}

InterfaceGroups {

    // Explicit member enumeration
    Group G1 { EquipmentInterface1, EquipmentInterface2 };

    // Implicit member enumeration based on attributes
    Group G2 { interfaces with LineName = "LineName" };
    // Implicit member enumerations can be combined with an And-filter
    Group G3 { interfaces with LineName = "LineName" and GrpInfo = "GrpInfoName" };

    // Combinations of explicit and implicit member definitions are also possible
    Group G4 { interfaces with LineName = "LineName", EquipmentInterface2 };

    // Interfaces can also excluded explicitly
    Group G5 { interfaces with LineName = "LineName" } without { EquipmentInterface1 };

    // Interfaces can also excluded implicitly
    Group G6 { interfaces with LineName = "LineName" } without { interfaces with PwSup1 = "PwSup1" };

    // Groups can be combined
    Group G7 combines G1 and G2 and G3;
}

/*
 * ADDITIONAL RESTRICTIONS
 */

 Restrictions {

    /*
     * DISLOCALITY
     */

    // EquipmentInterface1 and EquipmentInterface2 must
    // be on separate Compartments (and RDCs and Connectors)
    // (the level can be set to "Compartment" or "RDC" or "Connector")
    EquipmentInterface1, EquipmentInterface2 dislocal up to Compartment;

    // All members of G1 must be on separate RDCs
    G1 dislocal up to RDC;

    // The members of G1 and G2 must not share an RDC
    // (while the members of G1 may be on the same RDC)
    G1, G2 dislocal up to RDC;


    /*
     * COLOCALITY
     */

    // EquipmentInterface1 and EquipmentInterface2 must
    // be mapped to the same Connector (or RDC or Compartment)
    EquipmentInterface1, EquipmentInterface2 on same Connector;
    EquipmentInterface1, EquipmentInterface2 on same RDC;
    // This also works for groups
    G1 on same Connector;
    // .. and combinations of groups and interfaces
    G4, EquipmentInterface2, G5 on same RDC;

    /*
     * VALID DEPLOYMENTS
     */

    // Valid allocations for EquipmentInterface1 are all
    // pins on Connector1 of RDC1 in Comp1
    // (explicit pin specification)
    Valid for EquipmentInterface1 are { Comp1.RDC1.Connector1 };

    // This also works for groups
    // (explicit pin specification)
    Valid for G1, EquipmentInterface1, G3 are { Comp1 };

    // There are also implicit pin specifications possible
    Valid for EquipmentInterface1 are { connectors with Compartment.Name = "Comp1" };

    // Or even implicit combinations ...
    // EquipmentInterface1 can be allocated to all connectors
    // where the hosting RDC has a powerSupply1 attribute of Sup1 or
    // the hosting RDC has a powerSupply2 attribute of Sup2
    Valid for EquipmentInterface1 are { connectors with RDC.PowerSupply1 = "Sup1",
                                        connectors with RDC.PowerSupply2 = "Sup2" };

    /*
     * INVALID DEPLOYMENTS
     */

    Invalid for EquipmentInterface1 are { Comp1.RDC1.Connector1 };

    Invalid for G1, EquipmentInterface1, G3 are { Comp1 };

    Invalid for EquipmentInterface1 are { connectors with Compartment.Name = "Comp1" };

    Invalid for EquipmentInterface1 are { connectors with RDC.PowerSupply1 = "Sup1",
                                          connectors with RDC.PowerSupply2 = "Sup2" };

}
```

Listing A.1: **ASSIST Example Specification**

B. Input Specification Grammar

In Listing B.1 the grammar for the input specification language is depicted. Please refer to this Listing when a specification contains syntax errors.

```
grammar ch.hilbri . assist .mappingdsl.MappingDSL with org.eclipse.xtext.common.Terminals

import "ch.hilbri.assist.datamodel.model"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

AssistModel:
    (
        globalBlock             = GlobalBlock              &
        compartmentsBlock       = CompartmentsBlock        &
        interfacesBlock         =  InterfacesBlock         &
        (interfaceGroupsBlock   = InterfaceGroupsBlock)?   &
        ( restrictionsBlock     =  RestrictionsBlock )?
    )
;

/* ***********************************************************
 * GLOBAL BLOCK
 *********************************************************** */

GlobalBlock: {GlobalBlock}
        'Global' '{'
        (
            ('Name' '=' systemName=STRING ';'                        )?      &
            ( compatibleIoTypesBlock     = CompatibleIoTypesBlock     )?      &
            ( cableWeightDataBlock       = CableWeightDataBlock       )?      &
            ( protectionLevelDataBlock   = ProtectionLevelDataBlock   )?
        )
        '}'
;

CompatibleIoTypesBlock:
    'Compatible Interface Types' '{' (compatibleIoTypes+=CompatibleIoTypeEntry)+ '}'
;

CompatibleIoTypeEntry:
    eqInterfaceIoType=STRING '->' pinInterfaceIoTypes+=STRING (',' pinInterfaceIoTypes+=STRING)* ';'
```

```
;

CableWeightDataBlock:
    'Cable Weights'             '{' (cableWeightEntries+=CableWeightEntry)+    '}'
;

CableWeightEntry:
    (eqInterfaceIoType=STRING | defaultEntry?='default') '=' weight=Double ';'
;

ProtectionLevelDataBlock:
    'Protection Level Restrictions' '{' (protectionLevelEntries+=ProtectionLevelEntry)+ '}'
;

ProtectionLevelEntry :
    'RDC.Location' '=' rdcLocation=STRING 'And' 'Equipment.EmhZone1' '=' emhZone1=STRING '->
        ' protectionLevel+=ProtectionLevelType (',' protectionLevel+=ProtectionLevelType)* ';'
;


/* ************************************************************
 * COMPARTMENTS
 ************************************************************ */

CompartmentsBlock: {CompartmentsBlock}
    'Compartments' '{'
        compartments+=Compartment+
    '}'
;

Compartment:
    'Compartment' name=ID '{'
        (
            ('Manufacturer'         '=' manufacturer=STRING ';')?         &
            ('PowerSupply'          '=' powerSupply=STRING ';')?          &
            ('Side'                 '=' side=STRING ';')?                 &
            ('Zone'                 '=' zone=STRING ';')?                 &
            (rdcs+=RDC+)
        )
        (metricParametersBlock=MetricParametersBlock)?
    '}'
;

RDC:
    'RDC' name=ID '{'
        (
            ('Manufacturer'         '=' manufacturer=STRING ';')?         &
            ('PowerSupply1'         '=' powerSupply1=STRING ';')?         &
            ('PowerSupply2'         '=' powerSupply2=STRING ';')?         &
            ('Type'                  '=' rdcType=STRING ';')?                  &
            ('ESS'                   '=' ess=STRING ';')?                      &
            ('Location'              '=' location=STRING ';')?                 &
            ('ResourceX'             '=' resourceX=SIGNEDINT ';')?             &
            ('ResourceY'             '=' resourceY=SIGNEDINT ';')?             &
            ('ResourceZ'             '=' resourceZ=SIGNEDINT ';')?             &
            (internalConnectedPinBlock=InternalConnectedPinBlock)?             &
            (connectors+=Connector)+
        )
        (metricParametersBlock=MetricParametersBlock)?
    '}'
;

InternalConnectedPinBlock: {InternalConnectedPinBlock}
    'Connected Pins' '{'
        (connectedPins+=ConnectedPinEntry)*
    '}'
;

ConnectedPinEntry:
    pins+=[Pin|QualifiedName] ',' pins+=[Pin|QualifiedName] (',' pins+=[Pin|QualifiedName])?    'are connected' ';'
;

Connector:
    'Connector' name=ID '{'
        (pins+=Pin)*
        (metricParametersBlock=MetricParametersBlock)?
    '}'
;

Pin:
    name=ID ':' eqInterfaceType=STRING ('protection level' protectionLevel=ProtectionLevelType)? ';'
;


enum ProtectionLevelType:
    NONE  = 'None'  |
    L1    = 'L1'    |
    L2    = 'L2'    |
    L3    = 'L3'    |
    L4    = 'L4'    |
    L5    = 'L5'    |
    L6    = 'L6'    |
    L7    = 'L7'    |
    L8    = 'L8'
;

```

```
MetricParametersBlock: {MetricParametersBlock}
    ’Metric Parameters’ ’{’
        (metricParameters+=MetricParameter)*
    ’}’
;

MetricParameter:
    name=STRING ’=’ value=INT ’;’
;

/* ***********************************************************
 * INTERFACES
  *********************************************************** */

InterfacesBlock :
    ’Interfaces’ ’{’
        ( eqInterfaces+=EqInterface)+
    ’}’
;

EqInterface :
    ’Interface’ name=ID ’{’
        (
            (’System’        ’=’ system=STRING      ’;’)?          &
            (’SubAta’        ’=’ subAta=STRING      ’;’)?          &
            (’Resource’      ’=’ resource=STRING    ’;’)?          &
            (’LineName’      ’=’ lineName=STRING    ’;’)?          &
            (’WiringLane’    ’=’ wiringLane=STRING  ’;’)?          &
            (’GrpInfo’       ’=’ grpInfo=STRING     ’;’)?          &
            (’Route’         ’=’ route=STRING       ’;’)?          &
            (’PwSup1’        ’=’ pwSup1=STRING      ’;’)?          &
            (’EmhZone1’      ’=’ emhZone1=STRING    ’;’)?          &
            (’Type’          ’=’ ioType=STRING      ’;’)?          &
            (’ResourceX’     ’=’ resourceX=SIGNEDINT ’;’)?         &
            (’ResourceY’     ’=’ resourceY=SIGNEDINT ’;’)?         &
            (’ResourceZ’     ’=’ resourceZ=SIGNEDINT ’;’)?
        )
    ’}’;


/* ***********************************************************
 * INTERFACE GROUPS
 * *********************************************************** */

InterfaceGroupsBlock: {InterfaceGroupsBlock}
    ’InterfaceGroups’ ’{’
        (
            eqInterfaceGroups+=EqInterfaceGroup                      |
            eqInterfaceGroups+=EqInterfaceGroupWithCombinedDefinition
        )*
    '}'
;


EqInterfaceGroup:
    'Group' name=ID '{'
                    (
                        eqInterfaces+=[EqInterface] |
                        implicitMemberDefinitions+=ImplicitEqInterfaceMemberDefinition
                    )
                    (
                        ',' (
                                eqInterfaces+=[EqInterface] |
                                implicitMemberDefinitions+=ImplicitEqInterfaceMemberDefinition
                            )
                    )*

                    '}'

                (
                    'without' '{'
                    (
                        withoutEqInterfaces+=[EqInterface] |
                        withoutImplicitMemberDefinitions+=ImplicitEqInterfaceMemberDefinition
                    )
                    (
                        ',' (
                                withoutEqInterfaces+=[EqInterface] |
                                withoutImplicitMemberDefinitions+=
                                    ImplicitEqInterfaceMemberDefinition
                            )
                    )*

                    '}'
                )?

            ';';


ImplicitEqInterfaceMemberDefinition:
    'interfaces with' entries+=ImplicitEqInterfaceMemberDefinitionAttributesAndValues ('and' entries
        +=ImplicitEqInterfaceMemberDefinitionAttributesAndValues)*
;

ImplicitEqInterfaceMemberDefinitionAttributesAndValues:
    attribute=ImplicitEqInterfaceMemberDefinitionAttribute '=' value=STRING
;

```

```
enum ImplicitEqInterfaceMemberDefinitionAttribute:
    NAME        = 'Name'           |
    SYSTEM      = 'System'         |
    SUBATA      = 'SubAta'         |
    LINENAME    = 'LineName'       |
    WIRINGLANE  = 'WiringLane'     |
    GRPINFO     = 'GrpInfo'        |
    ROUTE       = 'Route'          |
    PWSUP1      = 'PwSup1'         |
    EMHZONE1    = 'EmhZone1'       |
    IOTYPE      = 'Type'           |
    RESOURCE    = 'Resource'       |
    RESOURCE_X  = 'ResourceX'      |
    RESOURCE_Y  = 'ResourceY'      |
    RESOURCE_Z  = 'ResourceZ'
;

EqInterfaceGroupWithCombinedDefinition:
    'Group' name=ID 'combines' combinedGroups+=[EqInterfaceGroup] ('and' combinedGroups+=[
        EqInterfaceGroup])+ ';'
;


/* ********************************************************
 * RESTRICTIONS
 * ******************************************************** */

 RestrictionsBlock : { RestrictionsBlock }
    'Restrictions' '{'
        (
            dislocalityRelations    +=  DislocalityRelation    |
            colocalityRelations     +=  ColocalityRelation     |
           validDeployments         += ValidDeployment         |
           invalidDeployments       += InvalidDeployment
        )*
    '}'
;

enum HardwareArchitectureLevelType :
    COMPARTMENT = 'Compartment' |
    RDC         = 'RDC'         |
    CONNECTOR = 'Connector'
;

 DislocalityRelation  :
    eqInterfaceOrGroups+=[EqInterfaceOrGroup]
    (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
    'dislocal up to'
    hardwareLevel=HardwareArchitectureLevelType ';';

 ColocalityRelation :
     eqInterfaceOrGroups+=[EqInterfaceOrGroup]
     (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
     'on same'
     hardwareLevel=HardwareArchitectureLevelType ';'
;

ValidDeployment:
     'Valid for' eqInterfaceOrGroups+=[EqInterfaceOrGroup]
     (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
     'are' '{'
             (          hardwareElements+=[HardwareElement|QualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition)
             (','  (hardwareElements+=[HardwareElement|QualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition) )*
     '}' ';'
;

InvalidDeployment:
     'Invalid for' eqInterfaceOrGroups+=[EqInterfaceOrGroup]
     (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
     'are' '{'
             (        hardwareElements+=[HardwareElement|QualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition)
             (',' (hardwareElements+=[HardwareElement|QualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition) )*
     '}' ';'
;

DeploymentImplicitDefinition :
     'connectors with' entries+=DeploymentImplicitDefinitionAttributeAndValue
     ('and' entries+=DeploymentImplicitDefinitionAttributeAndValue)*
;

DeploymentImplicitDefinitionAttributeAndValue:
    attribute = DeploymentImplicitDefinitionAttribute  '=' value=STRING
;

enum DeploymentImplicitDefinitionAttribute:
     COMPARTMENT_NAME          = 'Compartment.Name'           |
     COMPARTMENT_MANUFACTURER  = 'Compartment.Manufacturer'   |
     COMPARTMENT_POWERSUPPLY   = 'Compartment.PowerSupply'    |
     COMPARTMENT_SIDE          = 'Compartment.Side'           |
     COMPARTMENT_ZONE          = 'Compartment.Zone'           |

     RDC_NAME                  = 'RDC.Name'                   |
     RDC_MANUFACTURER          = 'RDC.Manufacturer'           |
    RDC_POWERSUPPLY1        = 'RDC.PowerSupply1'        |
    RDC_POWERSUPPLY2        = 'RDC.PowerSupply2'        |
    RDC_SIDE                = 'RDC.Side'                |
    RDC_TYPE                = 'RDC.Type'                |
    RDC_ESS                 = 'RDC.ESS'                 |
    RDC_RESOURCE_X          = 'RDC.ResourceX'           |
    RDC_RESOURCE_Y          = 'RDC.ResourceY'           |
    RDC_RESOURCE_Z          = 'RDC.ResourceZ'           |

    CONNECTOR_NAME          = 'Connector.Name'
;


/* ************************************************************
 * RESTRICTIONS
 * ************************************************************ */

Double returns ecore::EDouble:
    INT '.' INT
;

QualifiedName:
    ID ('.' ID)*
;

SIGNEDINT returns ecore::EInt:
  '-'? INT;
```

Listing B.1: **ASSIST Input Specification Grammar**