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	24th October 2015

Document History:

Version 0.1	Initial Version	18.08.2015
Version 0.2	Additional Structure	10.09.2015
Version 0.3	Introduction, Usage	28.09.2015

Contents

1. Introduction

The ASSIST Toolsuite is for System Engineers and System Architects of safety-critical systems. It automates the construction of a mapping between equipment interfaces and the pins of an RDC while respecting safety and reliability requirements.

The ASSIST Toolsuite aims to automate this challenging, error prone and complex task. It requires the user to specify:

- the resource requirements of the equipment interfaces,
- the features and capabilities of the RDCs and
- safety and reliability requirements

in a textual domain specific language.

This specification of a mapping problem is automatically transfered into Constraint Satisfaction Problem (CSP). Solutions for this CSP represent correct deployments. They are automatically generated with the constraint solver *Choco3*. Solutions can also be evaluated and optimized based on pre-defined or customizable metrics.

2. Installation and Usage

2.1. Installation

1. Download the zip archive and extract it to some folder. (The binary build already includes a suitable Java Runtime Environment, so you do not need to install a JRE separately.)

2. Start ASSIST by double clicking on ASSIST.exe (Windows only).

2.2. Usage

2.2.1. Overview

ASSIST has been developed to support the following workflow:

1. Import or specify your hardware and software architecture in an **mdsl** file

2. Import or Specify your resource and safety requirements in the same **mdsl** file

3. Generate a set of valid deployments

4. Iterative approach: partial solutions can be fixed in ASSIST

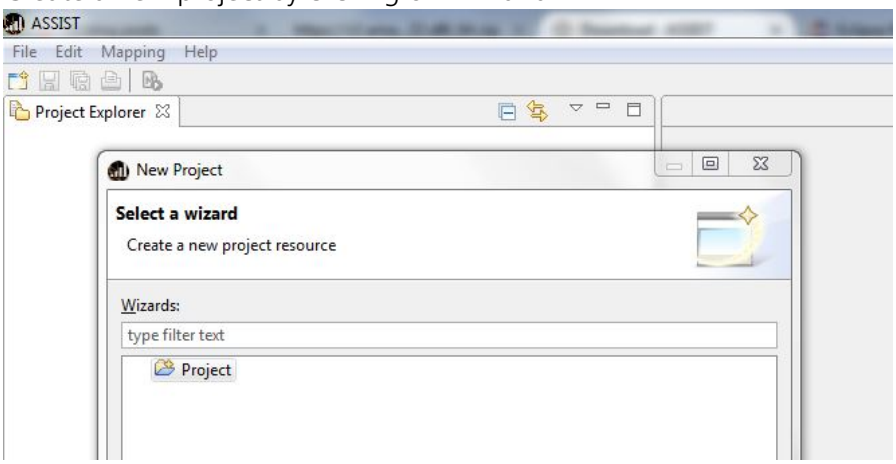
5. Sort these deployments according to customizable metrics

6. Select an optimal deployment

7. Export the selected deployment

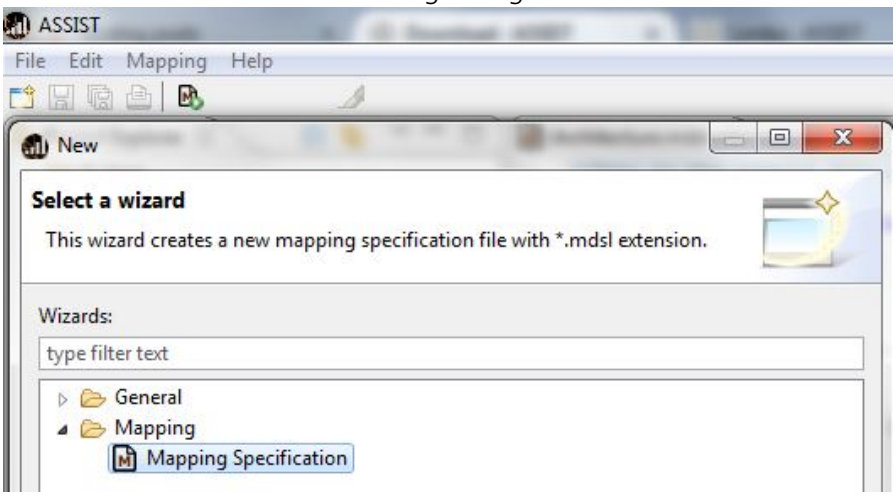
2.2.2. Create a new Project and Specification

1. Create a new project by clicking on FILE and NEW



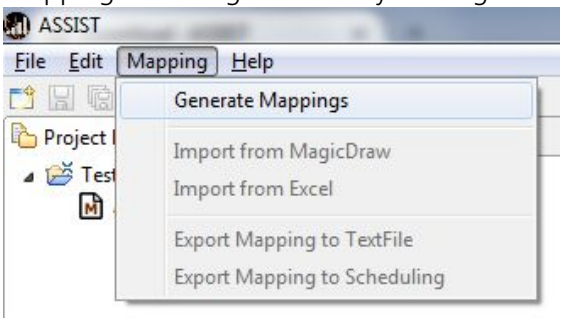
Please provide a suiteable name for your project and click on FINISH.

2. Create a new mapping specification by clicking on FILE and NEW. Please select MAPPING SPECIFICATION in the following dialog.



Please provide a suiteable name for your deployment specification and click on FINISH.

3. Mappings can be generated by clicking on MAPPING and GENERATE MAPPINGS



2.2.3. Generate Mappings

- Available Strategies
- Partial Solutions

2.2.4. Save Mappings to Specification

- How do I store individual mappings in ASIST?

2.2.5. Evaluate Solutions

- How do I evaluate and sort my solutions?
- Builtin metrics vs. customized metrics

2.2.6. Creating customized Metrics

- How do I create customized metrics?

2.3. Useful Shortcuts

- **Ctrl-Shift-C** - Comment the current line or the currently selected lines
- **Ctrl-Shift-F** - Format the current file (indention of sections, . . .)
- **Ctrl-Space** - Code-Completion; the current specification entry will be completed with available parameters
- **Ctrl-F** - Find and replace dialog
- **Ctrl-S** - Save the current editor file

3. System Elements

3.1. Overview

3.2. Compartments

3.3. RDCs

3.4. Pins

3.5. Equipment Interfaces

4. Constraint Semantics

4.1. Basic Interface Types Constraint

4.1.1. Intended semantics

4.1.2. Input Specification

4.1.3. Implementation Remarks

4.2. Configurable Interface Types Constraint

4.2.1. Intended semantics

4.2.2. Input Specification

4.2.3. Implementation Remarks

4.3. Protection Level Constrains

4.3.1. Intended semantics

4.3.2. Input Specification

4.3.3. Implementation Remarks

4.4. Connected Pins

4.4.1. Intended semantics

4.4.2. Input Specification

4.4.3. Implementation Remarks

4.5. Restriction: Colocality

4.5.1. Intended semantics

4.5.2. Input Specification

4.5.3. Implementation Remarks

4.6. Restriction: Dislocality

4.6.1. Intended semantics

4.6.2. Input Specification

4.6.3. Implementation Remarks

4.7. Restriction: Valid Deployments

4.7.1. Intended semantics

4.7.2. Input Specification

4.7.3. Implementation Remarks

4.8. Restriction: Invalid Deployments

4.8.1. Intended semantics

4.8.2. Input Specification

4.9. Inverse Channeling - Deployment from the Pin Perspective

4.9.1. Intended semantics

4.9.2. Input Specification

4.9.3. 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. Comments are used to further document the intended specification. Single-line comments are prefixed with `//`. Multi-line comments have to be placed between `/*` and `*/`. The entire specification language is white space tolerant, so that additional white spaces can be added between keywords without violating its syntax.

```
/*
 * 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";
33      Zone            = "ZoneName";

35      RDC RDC1 {
            Manufacturer    = "ManufacturerName";
37          PowerSupply1    = "PowerSupply1Name";
            PowerSupply2    = "PowerSupply2Name";
39          Type            = "RDCTypeName";
            ESS             = "ESSName";
41          Location        = "RDCLocatioName";
            ResourceX       = -120;
43          ResourceY       = 150;
            ResourceZ       = -1200;
45
            Connector Connector1{
47              // Available protection levels:
                // None, L1, L2, L3, L4, L5, L6, L7, L8
49
                Pin1: "PinType0"; // = protection level None
51              Pin2: "PinType0" protection level L5;
                Pin3: "PinType1" protection level L8;
53
            }
55
            Connector Connector2{
57              Pin1: "PinType3";
                Pin2: "PinType1" protection level L8;
59              Pin3: "PinType1" protection level L8;
            }
61
            // ... more Connectors possible ...
63
            Connected Pins {
65              Comp1.RDC1.Connector1.Pin1, Comp1.RDC1.Connector1.Pin2 are connected;
                Comp1.RDC1.Connector1.Pin3, Comp1.RDC1.Connector2.Pin2, Comp1.RDC1.Connector2
                    .Pin3 are connected;
67          }

69          Metric Parameters {
                "RDC1Parameter1" = 54;
71
                // ...
73          }
        }
75
        // ... more RDCs possible ...
77
        Metric Parameters {
79          "Comp1Parameter1" = 12;
```

```
            "Comp1Parameter2" = 6;
            // ...
        }
    }

    // Another empty compartment for demo purposes
    Compartment Comp2 {
        RDC RDC1 {
            Connector Connector1{
            }
        }
    }
}

/*
 * 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;
    }
```

```
129 }

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

135     // Implicit member enumeration based on attributes
        Group G2 { interfaces with LineName = "LineName" };
137
        // Implicit member enumerations can be combined with an And-filter
139     Group G3 { interfaces with LineName = "LineName" and GrpInfo = "GrpInfoName" };

141     // Combinations of explicit and implicit member definitions are also possible
        Group G4 { interfaces with LineName = "LineName", EquipmentInterface2 };
143
        // Interfaces can also excluded explicitly
145     Group G5 { interfaces with LineName = "LineName" } without { EquipmentInterface1 };

147     // Interfaces can also excluded implicitly
        Group G6 { interfaces with LineName = "LineName" } without { interfaces with PwSup1 = "PwSup1" };
149
        // Groups can be combined
151     Group G7 combines G1 and G2 and G3;
    }
153

155 /*
     * ADDITIONAL RESTRICTIONS
157  */

159  Restrictions {

161     /*
         * COLOCALITY
163      */

165     // EquipmentInterface1 and EquipmentInterface2 must
        // be mapped to the same Connector (or RDC or Compartment)
167
        EquipmentInterface1, EquipmentInterface2 on same Connector;
169     EquipmentInterface1, EquipmentInterface2 on same RDC;

171     // This also works for groups

173     G1 on same Connector;

175     // .. and combinations of groups and interfaces

177     G4, EquipmentInterface2, G5 on same RDC;
```

```
179     /*
         * DISLOCALITY
181      */

183     // EquipmentInterface1 and EquipmentInterface2 must
        // be on separate Compartments (and RDCs and Connectors)
185     // (the level can be set to "Compartment" or "RDC" or "Connector")

187 //  EquipmentInterface1, EquipmentInterface2 dislocal up to Connector;

189     // All members of G1 must be on separate Connector

191 //  G1 dislocal up to Connector;

193     // The members of G1 and G2 must not share an Connector
        // (while the members of G1 may be on the same Connector)
195
    //  G1, G2 dislocal up to Connector;
197


199     /*
         * VALID DEPLOYMENTS
201      */

203     // Valid allocations for EquipmentInterface1 are all
        // pins on Connector1 of RDC1 in Comp1
205     // (explicit pin specification)
        Valid for EquipmentInterface1 is { Comp1.RDC1.Connector1 };
207
        // This also works for groups
209     // (explicit pin specification)
        Valid for G1, EquipmentInterface1, G3 is { Comp1 };
211
        // There are also implicit pin specifications possible
213     Valid for EquipmentInterface1 is { pins with Compartment.Name = "Comp1" };

215     // Or even implicit combinations ...
        // EquipmentInterface1 can be allocated to all connectors
217     // where the hosting RDC has a powerSupply1 attribute of Sup1 or
        // the hosting RDC has a powerSupply2 attribute of Sup2
219     Valid for EquipmentInterface1 is { pins with RDC.PowerSupply1 = "PowerSupply1Name",
                                           pins with RDC.PowerSupply2 = "PowerSupply2Name" };
221
        /*
223      * INVALID DEPLOYMENTS
         */
225
        Invalid for EquipmentInterface1 is { Comp2.RDC1.Connector1 };
```

```
227
    Invalid for G1, EquipmentInterface1, G3 is { Comp2 };
229
    Invalid for EquipmentInterface1 is { pins with Compartment.Name = "UnsafeCompartment" };
231
    Invalid for EquipmentInterface1 is { pins with RDC.PowerSupply1 = "UnsafeSupplier",
233                                      pins with RDC.PowerSupply2 = "UnsafeSupplier" };

235 }
```

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)* ';'
```

```
37  ;

39  CableWeightDataBlock:
        'Cable' 'Weights'            '{' (cableWeightEntries+=CableWeightEntry)+    '}'
41  ;

43  CableWeightEntry:
        (eqInterfaceIoType=STRING | defaultEntry?='default') '=' weight=Double ';'
45  ;

47  ProtectionLevelDataBlock:
        'Protection' 'Level' 'Restrictions' '{' (protectionLevelEntries+=ProtectionLevelEntry)+ '}'
49  ;

51  ProtectionLevelEntry :
        'RDC.Location' '=' rdcLocation=STRING 'And' 'Equipment.EmhZone1' '=' emhZone1=STRING '->
             ' protectionLevel+=ProtectionLevelType (',' protectionLevel+=ProtectionLevelType)* ';'
53  ;

55

    /* ************************************************************
57   * COMPARTMENTS
       ************************************************************ */
59

    CompartmentsBlock: {CompartmentsBlock}
61      'Compartments' '{'
            compartments+=Compartment+
63      '}'
    ;
65

    Compartment:
67      'Compartment' name=ID '{'
            (
69              ('Manufacturer'          '=' manufacturer=STRING ';')?              &
                ('PowerSupply'           '=' powerSupply=STRING ';')?               &
71              ('Side'                  '=' side=STRING ';')?                      &
                ('Zone'                  '=' zone=STRING ';')?                      &
73              (rdcs+=RDC+)
            )
75          (metricParametersBlock=MetricParametersBlock)?
        '}'
77  ;

79  RDC:
        'RDC' name=ID '{'
81          (
                ('Manufacturer'          '=' manufacturer=STRING ';')?              &
83              ('PowerSupply1'          '=' powerSupply1=STRING ';')?              &
                ('PowerSupply2'          '=' powerSupply2=STRING ';')?              &
```

```
85              ('Type'                 '=' rdcType=STRING ';')?                &
                ('ESS'                  '=' ess=STRING ';')?                    &
87              ('Location'             '=' location=STRING ';')?               &
                ('ResourceX'            '=' resourceX=SIGNEDINT ';')?           &
89              ('ResourceY'            '=' resourceY=SIGNEDINT ';')?           &
                ('ResourceZ'            '=' resourceZ=SIGNEDINT ';')?           &
91              (internalConnectedPinBlock=InternalConnectedPinBlock)?          &
                (connectors+=Connector)+
93          )
            (metricParametersBlock=MetricParametersBlock)?
95      '}'
    ;
97
    InternalConnectedPinBlock: {InternalConnectedPinBlock}
99      'Connected' 'Pins' '{'
            (connectedPins+=ConnectedPinEntry)*
101     '}'
    ;
103
    ConnectedPinEntry:
105     pins+=[Pin|QualifiedName] ',' pins+=[Pin|QualifiedName] (',' pins+=[Pin|QualifiedName])? 'are'
            'connected' ';'
    ;
107
    Connector:
109     'Connector' name=ID '{'
            (pins+=Pin)*
111         (metricParametersBlock=MetricParametersBlock)?
        '}'
113 ;

115 Pin:
        name=ID ':' eqInterfaceType=STRING ('protection' 'level' protectionLevel=ProtectionLevelType)? '
            ;'
117 ;

119
    enum ProtectionLevelType:
121     NONE  = 'None'  |
        L1    = 'L1'    |
123     L2    = 'L2'    |
        L3    = 'L3'    |
125     L4    = 'L4'    |
        L5    = 'L5'    |
127     L6    = 'L6'    |
        L7    = 'L7'    |
129     L8    = 'L8'
    ;
131
```

```
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
;
```

```
229 enum ImplicitEqInterfaceMemberDefinitionAttribute:
        NAME       = 'Name'          |
231     SYSTEM     = 'System'        |
        SUBATA     = 'SubAta'        |
233     LINENAME   = 'LineName'      |
        WIRINGLANE = 'WiringLane'    |
235     GRPINFO    = 'GrpInfo'       |
        ROUTE      = 'Route'         |
237     PWSUP1     = 'PwSup1'        |
        EMHZONE1   = 'EmhZone1'      |
239     IOTYPE     = 'Type'          |
        RESOURCE   = 'Resource'      |
241     RESOURCE_X = 'ResourceX'     |
        RESOURCE_Y = 'ResourceY'     |
243     RESOURCE_Z = 'ResourceZ'
    ;
245
    EqInterfaceGroupWithCombinedDefinition:
247     'Group' name=ID 'combines' combinedGroups+=[EqInterfaceGroup] ('and' combinedGroups+=[
            EqInterfaceGroup])+ ';'
    ;
249

251 /* ************************************************************
     * RESTRICTIONS
253  * ************************************************************ */

255  RestrictionsBlock :  { RestrictionsBlock }
        'Restrictions' '{'
257         (
                dislocalityRelations      +=  DislocalityRelation          |
259             colocalityRelations       +=  ColocalityRelation           |
                validDeployments          += ValidDeployment               |
261             invalidDeployments        += InvalidDeployment
            )*
263     '}'
    ;
265
    enum HardwareArchitectureLevelType :
267     COMPARTMENT = 'Compartment' |
        RDC         = 'RDC'         |
269     CONNECTOR = 'Connector'
    ;
271
     DislocalityRelation  :
273     eqInterfaceOrGroups+=[EqInterfaceOrGroup] (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
        'dislocal' 'up' 'to' hardwareLevel=HardwareArchitectureLevelType ';';
275
     ColocalityRelation :
```

```
277     eqInterfaceOrGroups+=[EqInterfaceOrGroup] (',' eqInterfaceOrGroups+=[EqInterfaceOrGroup])*
        'on' 'same' hardwareLevel=HardwareArchitectureLevelType ';'
279 ;

281 ValidDeployment:
        'Valid' 'for' eqInterfaceOrGroups+=[EqInterfaceOrGroup] (',' eqInterfaceOrGroups+=[
            EqInterfaceOrGroup])* 'is'
283     '{'
                    (hardwareElements+=[HardwareElementlQualifiedName] | implicitHardwareElements+=
                        DeploymentImplicitDefinition)
285             (',' (hardwareElements+=[HardwareElementlQualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition) )*
        '}' ';'
287 ;

289 InvalidDeployment:
        'Invalid' 'for' eqInterfaceOrGroups+=[EqInterfaceOrGroup] (',' eqInterfaceOrGroups+=[
            EqInterfaceOrGroup])* 'is'
291     '{'
                (       hardwareElements+=[HardwareElementlQualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition)
293             (',' (hardwareElements+=[HardwareElementlQualifiedName] | implicitHardwareElements+=
                    DeploymentImplicitDefinition) )*
        '}' ';'
295 ;

297 DeploymentImplicitDefinition :
        'pins' 'with' entries+=DeploymentImplicitDefinitionAttributeAndValue
299     ('and' entries+=DeploymentImplicitDefinitionAttributeAndValue)*
    ;
301
    DeploymentImplicitDefinitionAttributeAndValue:
303     attribute = DeploymentImplicitDefinitionAttribute  '=' value=STRING
    ;
305
    enum DeploymentImplicitDefinitionAttribute:
307     COMPARTMENT_NAME      = 'Compartment.Name'        |
        COMPARTMENT_MANUFACTURER = 'Compartment.Manufacturer' |
309     COMPARTMENT_POWERSUPPLY = 'Compartment.PowerSupply' |
        COMPARTMENT_SIDE      = 'Compartment.Side'        |
311     COMPARTMENT_ZONE      = 'Compartment.Zone'        |
        RDC_NAME              = 'RDC.Name'                |
313     RDC_MANUFACTURER      = 'RDC.Manufacturer'        |
        RDC_POWERSUPPLY1      = 'RDC.PowerSupply1'        |
315     RDC_POWERSUPPLY2      = 'RDC.PowerSupply2'        |
        RDC_SIDE              = 'RDC.Side'                |
317     RDC_TYPE              = 'RDC.Type'                |
        RDC_ESS               = 'RDC.ESS'                 |
319     RDC_RESOURCE_X        = 'RDC.ResourceX'           |
```

```
    RDC_RESOURCE_Y          = 'RDC.ResourceY'          |
    RDC_RESOURCE_Z          = 'RDC.ResourceZ'          |
    CONNECTOR_NAME          = 'Connector.Name'         |
    PIN_NAME                = 'Pin.Name'
;

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

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

QualifiedName:
    ID ('.' ID)*
;

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

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