RosettaNet
EIPS/XSD Specification

Specification Information	
Name	RosettaNet EIPS/XSD Specification
Version	V11.00.00
Date	19 March 2008

Table of Content

List of Tables

List of Figures

1. Document Management

1.1. Legal Disclaimer

RosettaNet™, its members, officers, directors, employees, or agents MUST not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document or the specifications herein, as well as associated guidelines and schemas. The use of said specifications MUST constitute your express consent to the foregoing exculpation.

1.2. Copyright

1.3. Trademarks

RosettaNet, Partner Interface Process, PIP and the RosettaNet logo are trademarks or registered trademarks of "RosettaNet," a non-profit organization. All other product names and company logos mentioned herein are the trademarks of their respective owners. In the best effort, all terms mentioned in this document that are known to be trademarks or registered trademarks have been appropriately recognized in the first occurrence of the term.

1.4. Document Version History

Version	Date	Description
V11.00.00	March 19, 2008	Published as Validated specification

1.5. Related Documents

1) TPIR-PIP for Engineering Information Specification

1.6. Audience

This document's primary audience is Solution Providers and PIP Implementers that need to create and use EIPS/XSD(s) to be inserted in RosettaNet released PIP(s) with entry-points (extension points) for such additional information in order to better support their business processes. The secondary audience is RosettaNet engineers who maintain the community agreed EIPS/XSD(s).

1.7. Scope

The content of this document covers the data model, structure, and content coding for an EIPS/XSD.

1.8. Document Conventions

The keywords MUST, MUST NOT, REQUIRED, MUST, MUST NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119] as quoted here:

MUST This word, or the terms "REQUIRED" or "MUST", means that the definition is an absolute requirement of the specification.

MUST NOT *This phrase, or the* phrase *"MUST NOT", means that the definition is an absolute prohibition of the specification.*

SHOULD *This word, or the* adjective *"RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.*

SHOULD NOT *This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular* circumstances *when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.*

MAY *This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the* option*, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).*

1.9. Document Structure

This document includes a detailed description of rules and representation of an EIPS/XSD.

1.10. Acknowledgement

This document has been prepared by the EIM Foundational Program TPIR-PIP for Engineering Information prototyping team consisting of voluntary RosettaNet Japan members, with significant help from the EIM Milestone Program participants, RosettaNet Global architecture team and RosettaNet members.

2. Overview

2.1. Basic Concepts

Technical dictionaries are repositories of information about products. To ensure that there is enough information about a given product, a mass of information is defined. RosettaNet has the RNTD (in DTD format) as its technical dictionary.

The RNTD has a function for grouping characteristics called “property definition sets” (PDS). A PDS lists the codes assigned to Properties and exists as an independent item in the RNTD.

Example of an actual Property Definition Set in the RNTD:

```
<PropertyDefinitionSet
    id="EIM001-001"
    propDefs="
        RNP2048-001
        RNP-XJH259-001
        RNP-XJH261-001
        XJE219
        XJE417
        XJE418
        XJF102
        XJH250-001
        XJH251-001
        XJH254-001
        XJH260-001
        XJH262-001
        XJH268-001
        XJH269-001
        XJH460-001
        XJH727-001">
    <identifiers>
        <code>EIM001</code>
        <majRev>001</majRev>
        <minRev>001</minRev>
        <date.def>2006-03-17</date.def>
    </identifiers>
    <names>
        <preferred.name>EIM Property Set for A/D-D/A Convertors</preferred.name>
    </names>
    <definition.short>
        The community-agreed subset of properties for A/D-D/A convertors.
    </definition.short>
    <remark>This property set is based in part on ECALS class XJA685.</remark>
    <app.specific name="industry.domains">EC</app.specific>
</PropertyDefinitionSet>
```

Due to the massive size of the RNTD, partners frequently ask that RosettaNet to provide subsets of the Technical Dictionary.

An EIPS realizes the concept of a community-agreed subset of all the characteristics for a given product class or a given engineering information set. This document describes how such a concept should be implemented in XML schema, as an EIPS/XSD.

2.2. Data Model Requirements

Requirement 2-1: Every item must be uniquely identified by identifiers.

Rationale: Non-unique identifiers can cause parsing errors. The name and/or code (plus major revision) are typically used as the key for backend mapping so both should be kept unique to prevent errors in the business processing side too.

Requirement 2-2: Every item must provide sufficient semantics to aid in mapping data into backend systems.

Rationale: Data is provided in a message instance in order to be stored in a back-end system. That data will be useless unless it can be correctly mapped into a given back-end system. The identifiers uniquely identify EIPS/XSD and Characteristics. The Semantics provide information necessary to correctly map the contents of the value to a specific place in a backend database. The values provide information on the value type and its required format in support of error-checking the value.

2.2.1. Identifiers

Table 2.1 lists the identifiers. A combination of the Name, Code, and MajorRevision will uniquely identify the item. The other data items are provided for human recognition of the kind of change and the timing of the change. This combination of identification data has proven to be the minimum amount necessary over five years of actual messaging experience.

Name	**Meaning**	**Cardinality**
Name	Preferred name for the item. This name must be unique within the range where it is defined.	1
Code	A code that uniquely identifies this item within the namespace to which it belongs	0..1
MajorRevisionNumber	A number that is incremented by 1 each time this item is revised in such a way as to affect mappings using this item. Required when the Code is defined.	0..1
MinorRevisionNumber	A number that is incremented by 1 each time this item is revised in a way that does not affect mappings using this item and is reset to 0 when a major revision occurs	0..1
DateCreated	Date on which this item was created	1
DateOfLastMajorRevision	Date that this item last had a major revision	1
DateOfLastMinorRevision	Date that this item last had a minor revision	0..1

Table 2.1 Identifiers for Technical Data Objects

2.2.2. Semantic Information

Table 2.2 lists the semantic information. Semantic information supports the accurate mapping of the data to a backend system. This amount of data items has proven, over five years of actual messaging, to be the minimum amount necessary to support accurate backend mapping.

Name	**Description**	**Cardinality**
Alphanumeric Symbol	The preferred symbol represented in 8-bit ascii code	0..1
Definition	A combination of definitions and remarks to aid in mapping the item	1
Reference Source	An identification of the source of the name and definition if it is obtained from another source, such as the IEC dictionary or the ECALS dictionary	0..n
Graphic	A line drawing that describes the item	0..n

Table 2.2 Semantic Information for Technical Data Objects

2.2.3. Value Information

Requirement 2-3: Every Characteristic MUST contain a value.

Rationale: A null value in a message just wastes communication bandwidth.

Requirement 2-4: To support error-checking, each data value included in a message instance MUST be defined by a data type and the format for that data type.

Rationale: Technical information databases are a relatively recent development and their content has not been validated as much as the content in business databases, resulting in a noticeable amount of invalid data. Furthermore, only the largest suppliers already have databases in place, whereas most SME(s) keep their technical data either in electronic document or in a paper form and re-enter the data for RosettaNet messages.

Requirement 2-5: The definition for that value MUST also define the units for that value where possible.

Rationale: If the dictionary defines the units, conversion between units (such as inches to centimeters) becomes unnecessary and the message need not carry that information.

There are two basic types of values provided for these characteristics: measured values and defining/identifying values. The defining values or identifying values can be specified much like business values. But measured values are more complicated for many reasons:

1. Some characteristics have measured values that fall within a certain range of values. The highest value for this range is called the maximum value (max), the lowest value for this range is called the minimum value (min), the most common value is called the typical value (typ), the value the designers expected is called the specification value (spec), and the value that defines this class of products (gives it its differentiating name) is called the nominal value (nom).
2. Another way to describe a range of measured values is to show the values as a tolerance range around a nominal value. This is standard practice in describing physical measurements such as the size of holes in a printed circuit board or the spacing of pins on a chip, but it is also used for certain electrical characteristics such as resistance.
3. Some measured values are defined in an expression based on other measured values.

Name	**Description**	**Cardinality**
DataType	One of Integer, Real, Float, Boolean, String, Date, or Code List	1
ValueFormat	Format the value MUST be represented in	1
Units	The “Units of Measure” for a measured data item	0..1

Table 2.3 Information Needed to Describe Values

2.2.4. RosettaNet Design Practice

Requirement 2-6: The EIPS/XSD coding SHOULD match current RosettaNet practice wherever possible.

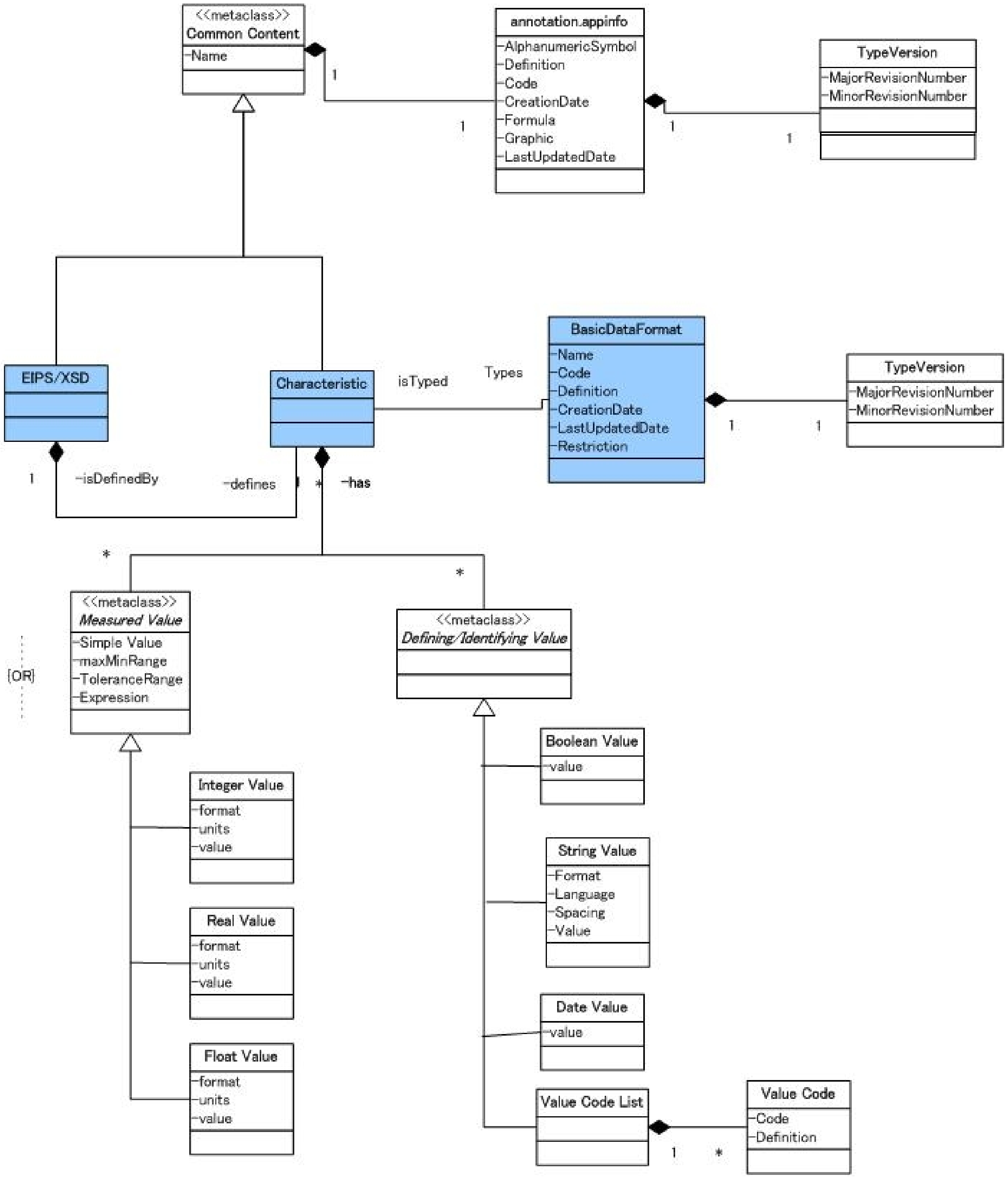
Rationale: Compatibility reduces system development costs.

Another source of data model requirements is current RosettaNet practice. Currently the minimum requirements for a RosettaNet data object is that it consist of the following tags:

```
<annotation>
    <appinfo>
        <urss:CreationDate>[DateCreated]</urss:CreationDate>
        <urss:Definition>[Definitions][Remark]</urss:Definition>
        <urss:LastUpdatedDate> Latest of either [DateOfLastMajorRevision] or [DateOfLastMinorRevision]</urss:LastUpdatedDate>
        <urss:TypeVersion>[MajorRevisionNumber].[MinorRevisionNumber]</urss:TypeVersion>
    </appinfo>
</annotation>
```

3. Data Model

Given the requirements described in chapter 2, the data model for an EIPS/XSD MUST be as shown in the Figure 3.1.



Detailed Implementation Model of EIPS data requirements (based on RNBD xsd coding requirements)

Figure 3.1 Detailed Implementation Model

Block Name	Contents
Common Content	The container for data elements that are common to an EIPS and its contained Characteristics.
Annotation.appinfo	The container that contains computer-processable data items in compliance with RosettaNet current practice.
Definition	The container that contains definitional information and remarks, following RosettaNet current practice.
TypeVersion	The container that contains a 2-digit MajorRevisionNumber followed by a period followed by a 2-digit MinorRevisionNumber.
EIPS/XSD	Engineering Information Property Set, a container (in xsd format) of a set of engineering information, which is the community-agreed subset of the full set of characteristics for a specific piece of technical information. Besides the Common Elements, it consists of one or more Characteristics.
Characteristic	A container for information on technical values. Besides the Common Elements, it contains a value that is restricted by a Basic Data Format.
Basic Data Format	A container for information on a format restriction for data values.
Measured Value	An abstract concept that groups together data values that have been measured and can be used in computations. MeasuredValues may be simple values, min/max values ranges or tolerance value ranges.
Defining/Identifying Values	An abstract concept that groups together data values that describe or identify.
Integer Value	The container for information on measured values that are expressed as an integer.
Real Value	The container for information on measured values that are expressed as a real value, that is, one containing a decimal point and a fraction.
Float Value	The container for information on measured values that are expressed as an floating point number, that is, one containing a fraction and an exponent.
Boolean Value	The container for information on boolean values, that is, values such as yes/no or true/false.
String Value	The container for information on character string values.
Date Value	The container for information on dates.
Value Code List	The container for information on a set of fixed codes, such as the abbreviations for kinds of memory (RAM, ROM, DRAM, DDRAM, etc).
Value Code	The container for information on a single code in a value code list.

Table 2.4 Data Model Classes and Descriptions

This data model defines three basic data objects, the EIPS/XSD, the Characteristic, and the Basic Data Format. All other data structures in the data model support the creation of these three basic data objects.

The data model also shows that a large amount of data items are common to these three basic data objects.

4. Content Structure Rules

This chapter provides the following information in the following sequence:

1. Information related to the Common Content data elements
2. Information related to Basic Data Format objects
3. Information related to Characteristic objects
4. Information related to EIPS/XSD objects

Note: the coding conventions used in all the following examples are

1. Text in black is to be coded exactly as shown.
2. Text in red, enclosed in square brackets is to be replaced by text with that meaning.
3. Text in blue indicates an action.

4.1. Common Data elements

As described in section 2.2.4 above, currently the minimum requirements for a RosettaNet data object is that it consistent with the following tags:

```
<element or type name="[Name]">
   <xs:annotation>
      <xs:appinfo>
         <urss:CreationDate>[DateCreated]</urss:CreationDate>
         <urss:Definition>[ShortDefinition][ExtendedDefinition][Remark]</urss:Definition>
         <urss:LastUpdatedDate>Latest of either [DateOfLatestMajorRevision] or
                                   [DateOfLatestMinorRevision]</urss:LastUpdatedDate>
         <urss:TypeVersion>[MajorRevisionNumber].[MajorRevisionNumber]</urss:TypeVersion>
      </xs:appinfo>
   </xs:annotation>
</element or type>
```

Rule 4-1: All the data objects in an EIPS/XSD MUST include the tags minimally required by RosettaNet practice.

Rationale: Requirement 2.6.

Rule 4-2: All the data objects in an EIPS/XSD MUST include the data items shown in Table 4.1 when their cardinality is 1 or more and MAY include those optional items when the partner's data requirements specify that they are required.

Rationale: Requirements 2.1 and 2.2.

Name	**Meaning**	**Cardinality**
Name	Preferred name for the item	1
Code	A code that uniquely identifies this item within the namespace to which it belongs	0..1
DateCreated	Date on which this item was created	1
DateOfLastMajorRevision	Date that this item last had a major revision	1
DateOfLastMinorRevision	Date that this item last had a minor revision	0..1
AlphanumericSymbol	The preferred symbol represented in 8-bit ascii code	0..1
ShortDefinition	A brief noun phrase that defines the item	1
ExtendedDefinition	Further normative information that clarifies the short definition	0..1
ReferenceSource	An identification of the source of the name and definition if it is obtained from another source, such as the IEC dictionary or the ECALS dictionary	0..n
Remark	Non-normative additional information in support of mapping, such as why this item should be used for a given concept rather than a similar item	0..1
Graphic	A line drawing that describes the item	0..n
MajorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in such a way as to affect mappings using this item. Required when the Code is defined.	1
MinorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in a way that does not affect mappings using this item and is reset to 0 when a major revision occurs	1
ValueFormat	The format of the item	0..1
Units	The “Units of Measure” for a measured data item	0..1

Table 4.1 Date Items that are Common to All Items in an EIPS

Rule 4-3: To satisfy rule 4-1 and 4-2, the XSD coding MUST include the following common items in the appinfo section of the annotation section:

urss:CreationDate,

urss:Definition,

urss:LastUpdatedDate,

urss:TypeVersion.

Rationale: Although these items are primarily intended for human recognition (to assist in the mapping of the information to a backend system), a solution provider may want to be able to access them for value-added processing, thus they have to be in the appinfo section.

Example:

```
<xs:annotation>
    <xs:appinfo>
        <urss:CreationDate>[DateCreated]</urss:CreationDate>
        <urss:Definition>[ShortDefinition][ExtendedDefinition][Remark]</urss:Definition>
        <urss:LastUpdatedDate>Latest of either [DateOfLatestMajorRevision] or
                                      [DateOfLatestMinorRevision]</urss:LastUpdatedDate>
        <urss:TypeVersion>[MajorRevisionNumber].[MinorRevisionNumber]</urss:TypeVersion>
    </xs:appinfo>
</xs:annotation>
```

Rule 4-4: The semantic information for a technical data item MUST allow for the optional coding of an AlphanumericSymbol, Graphic, and multiple normative and non-normative Definition content.

Rationale: A Definition alone may not be sufficient or efficient for fast mapping to backend information. For many engineers the alphanumeric symbol has greater recognition value while for other Characteristics a graphic may be necessary. At the same time, the concepts of a short definition, extended definition, and remark are standard practice in technical dictionaries (they are defined in IEC 61360) while RosettaNet does not distinguish the concepts. To maintain reasonable compatibility with both practices, the short definition, extended definition, and remark have to be combined.

4.2. Basic Data Formats

Rule 4-5: All data items MUST be checked for format and perhaps also for value range.

Rationale: Requirement 2-4.

Rule 4-6: The formats MUST be standardized, using an external set of standard formats.

Rationale: One of the most important purposes of an EIPS/XSD is to improve error checking of technical content To make the definition of common data formats easier and to minimize coding errors when defining characteristics, an EIPS/XSD uses the concept of “basic data types”. The use of these standard formats will make the creation of new EIPS/XSDs easier and also facilitate the definition of the related fields in the back-end database.

The value formats use the rules in ISO 6093 and IEC 61360 to describe value formats in a human readable format. The following table shows the value formats currently being used for RosettaNet technical data and their meaning.

Value Format	**Meaning**
B 1	Boolean value (True/False or 1/0)
M..10	Alphanumeric data with a max length of 10 characters
M..20	Alphanumeric data with a max length of 20 characters
M..32	Alphanumeric data with a max length of 32 characters
M..60	Alphanumeric data with a max length of 60 characters
M..100	Alphanumeric data with a max length of 100 characters
M..240	Alphanumeric data with a max length of 240 characters
M..255	Alphanumeric data with a max length of 255 characters
M..2048	Alphanumeric data with a max length of 2048 characters
N 9	Special code consisting of exactly 9 digits
N 4	Special code consisting of exactly 4 digits
N..10	Special code consisting of up to 10 digits
N 14	Special code consisting of exactly 14 digits
NR1 S..10	Signed integer value with up to 10 digits, used in calculations
NR2..2.3	Unsigned real value (contains decimal point but no exponent) with up to 10 digits, used in calculations
NR3 S..3.3ES2	Signed floating value with up to 6 digits and an exponent, used in calculations
NR3 S..7.7ES2	Signed floating value with up to 14 digits and an exponent, used in calculations
NR3..3.3ES2	Unsigned floating value with up to 6 digits and an exponent, used in calculations
X 16	Date

Table 4.2 Required Value Formats and Meaning

Note: These basic data types are not exactly the same as the RosettaNet universal data items like DUNS or DUNS plus 4. The universal data items are named data objects with a fixed format of some kind linked to that data object. On the other hand, the basic data formats are fixed formats that are shared by many different named data objects. The purpose of using such shared data formats is to reduce coding errors when defining a new XML schema object.

Because a Basic Data Format is just a restriction on a real Characteristic, some of the common data elements do not apply and can be deleted from the structure of the Basic Data Format. The necessary items are listed in the following table.

Name	**Meaning**	**Cardinality**
Name	Preferred name for the item.	1
Code	A code that uniquely identifies this item within the namespace to which it belongs.	1
DateCreated	Date on which this item was created.	1
ShortDefinition	A short definition of the basic data type.	1
DateOfLastMajorRevision	Date that this item last had a major revision.	1
DateOfLastMinorRevision	Date that this item last had a minor revision.	0..1
Restriction	A general expression that is used to validate the coding of a value in a message.	1
MajorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in a way that does not affect mappings using this item and is reset to 0 when a major revision occurs.	1
MinorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in a way that does not affect mappings using this item and is reset to 0 when a minor revision occurs.	1
ValueFormat	A format that matches the meaning of the Name and is written according to the rules ISO 6093 and IEC 61360.	1

Table 4.3 Data Fields Needed for a Basic Data Format

Rule 4-7: Each of the basic data types MUST consist of the following tags: xs:simpleType, xs:annotation, xs:appinfo, urss:Code, urss:CreationDate, urss:Definition, urss:LastUpdatedDate, urss:TypeVersion, urss:ValueFormat, xs:restriction.

Rationale: Because a Basic Data Format is just a restriction on a real Characteristic, some of the common data elements do not apply and can be deleted from the structure of the Basic Data Format.

Example:

<xs:simpleType name="[Name]">

<xs:annotation>

<xs:appinfo>

<urss:Code>[Code]</urss:Code>

<urss:CreationDate>[DateCreated]</urss:CreationDate>

<urss:Definition>[ShortDefinition]</urss:Definition>

<urss:LastUpdatedDate> Latest of either [DateOfLastMajorRevision] or [DateOfLastMinorRevision]</urss:LastUpdatedDate>

<urss:TypeVersion>[MajorRevisionNumber].[MinorRevisionNumber]</urss:TypeVersion>

<urss:ValueFormat>[ValueFormat]</urss:ValueFormat>

</xs:appinfo>

</xs:annotation>

<xs:restriction base="xs:string">

```
<xs:pattern value="[Select from the validation general expression column of Table 6.3 in accordance with contents of ValueFormat]"/ >

</xs:restriction>

</xs:simpleType>
```

Data items needed specifically to define the format of a Basic Data Format are listed in the following table.

Name	Meaning	Cardinality
ValueFormat	Format selected for the value column of Table 5.3.	1
Pattern	Selected from the validation general expression column of Table 5.3 in accordance with the above ValueFormat	1

Table 4.4 Data Items Needed Specifically to Define a Basic Data Format

Rule 4-8: The content of xs:pattern field MUST be based on the basic type “xs:string” and MUST be the “general expression” pattern in Table 5.3 that corresponds to the format in the ValueFormat field.

Rationale: If the correct general expression pattern is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.

Note: This information is provided for the reader’s understanding only. Because these basic data types are shared across all EIPS/XSDs, RosettaNet will provide and maintain these basic data types.

4.3. Characteristics

The structure of a characteristic depends on the type of data that it is defining. There are three major kinds of data types that result in different coding structures:

1. Unmeasured data types that are not value code lists
2. Unmeasured data types that are value code lists
3. Measured data types

4.3.1. Unmeasured Data Types that are not value code lists

The unmeasured (Simple) data types that are not value code lists are Boolean, String, and Dates. In this section, we define the structure rules for these three since they utilize the same structure. The structure for Value Code Lists is defined in the next section. None of these data types utilize Units of Measure. Furthermore, Boolean and Date use xml basic types and thus do not need a ValueFormat.

Rule 4-9: The standardized structure for simple, unmeasured data types (that are not value codes and that are not reused) MUST consist of the following codes: xs:element, xs:annotation, urss:ValueFormat.

Rationale: Simple data elements that are not reused are declared using xs:element according to RosettaNet practice. The xs:annotation tag is necessary to introduce the Common Content, and the urss:ValueFormat is necessary for the human who maps this data to the backend system to understand what the format of the item will be.

Rule 4-10: The xs:element tag MUST define its type attribute as the basic data type listed in Table 5.3 that corresponds to the content of ValueFormat.

Rationale: Using one of the basic data types will reduce the possibility of a coding error.

Example:

```
<xs:element name="[Name]" type=”[NameOfBasicDataFormat]”>
<xs:annotation>
<xs:appinfo>
{Coding for the common parts of the appinfo structure is defined in rule 4-3}
<urss:ValueFormat>[ValueFormat]</urss:ValueFormat>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

Name	**Meaning**	**Cardinality**
Name	Preferred name for the item	1
NameOfBasicDataFormat	Name selected from the name column of Table 5.3	1
ValueFormat	Description of format using IEC 61360-style format selected from the value column of Table 5.3	1

Table 4.5 Minimum Information Needed for a Simple Data Item Other Than the Common Content

Example:

```
<xs:element name="ProductMarking" type=”zebf:EIPSString255BasicType”>
<xs:annotation>
<xs:appinfo>
{Coding for the common parts of the appinfo structure is defined in rule 4-3}
<urss:ValueFormat>M..255</urss:ValueFormat>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

4.3.2. Unmeasured Data Types that are Value Code Lists

When the property specifies a value code type (a string that is part of a fixed set of strings, such as the 2-letter abbreviations for the states in the U.S.A.), the coding is almost identical as that for the simple data types, except that list of codes is defined in the element itself and there is not reference to an external basic format. RosettaNet practice requires each value code list to be defined in a separate file. See the RosettaNet XML Specification for the rules for making value code lists.

Rule 4-11: The value code list MUST follow RosettaNet design rules for defining code lists, as well as including the common elements. That means that the value code list MUST be specified in a separate file stored in the directory Code Lists under the Design directory. If the EIPS/XSD needs any of these value code lists, they MUST be imported.

Rationale: Requirement 2.6

Example of standard format:

```
< xs:element ref="[NamespacePrefix][CodeListName]" />
```

Actual Example:

```
<xs:element ref="dcadff:CADFileFormat" minOccurs="0"/>
```

Note: Value Code Lists do not require the following common fields:

- AlphanumericSymbol: code lists are not identified with an alphanumeric symbol
- Graphic: code lists never have a graphic
- ReferenceSource: the reference source is identified by a different set of tags
- ValueFormat: the code list format is as defined by RosettaNet design rules and is in effect self-defining.

Name	**Meaning**
NameSpacePrefix	The namespace prefix for the CodeList.
CodeListName	The name for the CodeList.

Table 4-6 Minimum Information Needed for a CodeList Other Than the Common Content

4.3.3. Measured Data Types

Measured data types always have an IEC ValueFormat beginning with NR; that is, they will be NR1, NR2, or NR3 values.

There are three basic kinds of technical data: simple values, toleranced values, and values with minimum, maximum, typical, nominal, and/or spec values. Any technical data transferred in a PIP must be expressed as one of those three “types”. The data model for this data requirement is shown in Figure 4.1.

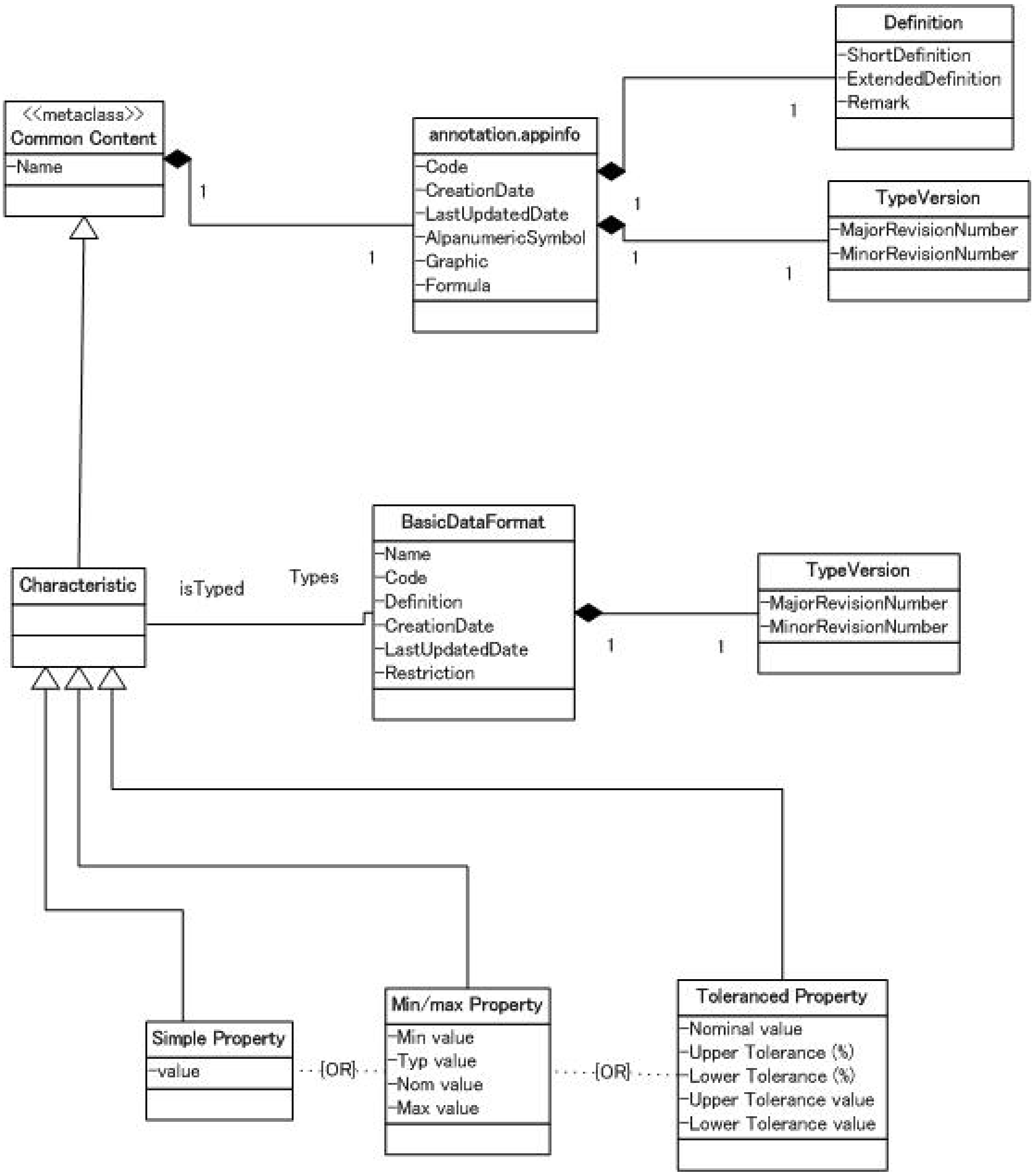


Figure 4.1 Data Model for Measured Values

The key point of this diagram is that a measured value has exactly the same data requirements as any other characteristic, except that it must allow a choice between a simple value or two different structures that, in turn, consist of sets of simple values.

Min/max characteristics and toleranced characteristics apply only to characteristics that are measured, that is, have a data format of integer, real, or float in the data model. The ability to enable the exchange of multiple values exists only for this kind of property and does not apply to any other property in the data model.

Rule 4-12: The coding for a value that may potentially be any of the three structures MUST be a choice between the three structures, unless the required structure is defined in the definition of the Characteristic.

Rationale: Normally, when the EIPS/XSD is being defined, the definers will not be able to determine which of the three structures the partners will choose. However, the definers should look at the definition of the characteristic because sometimes it will state that a given characteristic is expected to be exchanged with a certain set of values, such as min, typ, and max.

Rule 4-13: Because the coding uses a choice structure, the structure becomes a complexType. This complexType MUST be declared globally within the file containing it.

Rationale: This complies with current RosettaNet Practice.

Example:

Note that the “appinfo” section is the same as for a simple, non-measured value. The “choice” section will be modified, as required, during the restriction process.

```
<xs:complexType name="[Name]Type">
<xs:annotation>
<xs:appinfo>
{Coding for the common parts of the appinfo structure is defined in rule 4-3}
<urss:ValueFormat>[ValueFormat]</urss:ValueFormat>
</xs:appinfo>
</xs:annotation>
<xs:sequence>
<xs:choice>
        <!-- Simple property-->
        <xs:element name="ActualValue" type="[NameOfBasicDataFormat]"/>
        <!-- nominal property with optional tolerance-->
        <xs:sequence>
        <xs:element name="Nominal" type="[NameOfBasicDataFormat]"/>
        <xs:choice>
                <xs:sequence>
                        <xs:element name="LowerAbsoluteTolerance"
                                        type="[NameOfBasicDataFormat]" minOccurs="0"/>
                        <xs:element name="UpperAbsoluteTolerance"
                                        type="[NameOfBasicDataFormat]" minOccurs="0"/>
                </xs:sequence>
                <xs:sequence>
                        <xs:element name="LowerPercentTolerance"
                                        type="xs:decimal" minOccurs="0"/>
                        <xs:element name="UpperPercentTolerance"
                                        type="xs:decimal" minOccurs="0"/>
                </xs:sequence>
        </xs:choice>
        </xs:sequence>
```

```
    <!-- max/min property (main property must be typ or nom) -->
    <xs:sequence>
    <xs:element name="Max" type="[NameOfBasicDataFormat]" minOccurs="0"/>
    <xs:element name="Min" type="[NameOfBasicDataFormat]" minOccurs="0"/>
    <xs:element name="Nom" type="[NameOfBasicDataFormat]" minOccurs="0"/>
    <xs:element name="Typ" type="[NameOfBasicDataFormat]" minOccurs="0"/>
    </xs:sequence>
</xs:choice>
</xs:sequence>
</xs:complexType>
```

Name	**Meaning**
Name	Preferred name for the item.
ValueFormat	Description of format using IEC 61360-style format selected from the value column of Table 6.3.
NameOfBasicDataFormat	Name selected from the name column of Table 6.3.

Table 4.7 Minimum Information Needed for a Measured Value Other Than the Common Content

Example:

```
<xs:complexType name="TangentOfLossAngleType">
<xs:annotation>
<xs:appinfo>
{Coding for the common parts of the appinfo structure is defined in rule 4-3}
<urss:ValueFormat>NR3 S..3.3ES2</urss:ValueFormat>
</xs:appinfo>
</xs:annotation>
<xs:sequence>
<xs:choice>

    <!-- Simple property-->
    <xs:element name="ActualValue" type="zebf:EIPSSignedFloat6BasicType"/>

    <!-- nominal property with optional tolerance-->
    <xs:sequence>
    <xs:element name="Nominal" type="zebf:EIPSSignedFloat6BasicType"/>
    <xs:choice>
        <xs:sequence>
            <xs:element name="LowerAbsoluteTolerance"
                    type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
            <xs:element name="UpperAbsoluteTolerance"
```

```
                                type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
                </xs:sequence>
                <xs:sequence>
                        <xs:element name="LowerPercentTolerance"
                                type="xs:decimal" minOccurs="0"/>
                        <xs:element name="UpperPercentTolerance"
                                type="xs:decimal" minOccurs="0"/>
                </xs:sequence>
        </xs:choice>
        </xs:sequence>

        <!-- max/min property (main property must be typ or nom) -->
        <xs:sequence>
        <xs:element name="Max" type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
        <xs:element name="Min" type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
        <xs:element name="Nom" type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
        <xs:element name="Typ" type="zebf:EIPSSignedFloat6BasicType" minOccurs="0"/>
        </xs:sequence>
</xs:choice>
</xs:sequence>
</xs:complexType>
```

4.4. EIPS/XSD Structure

An EIPS/XSD in the data model maps to a RosettaNet structure. To be accessed from another schema it is defined as a complexType, just like ordinary RosettaNet structures with the exception of the additional identification and semantic information required from the data model.

Rule 4-14: The standardized structure for Property Definition Set types MUST be a complexType with the common elements declared above and with the element declarations for all the Characteristics needed for the Property Definition Set.

Rationale: This is standard RosettaNet practice.

Example:

```
<xs:complexType name="[EIPSName]Type">
<xs:annotation>
<xs:appinfo>
{Coding for the common parts of the appinfo structure is defined in rule 4-3}
</xs:appinfo>
</xs:annotation>
        <xs:sequence>
        <xs:element name="[Name]"  type="[Name]Type" minOccurs="0"/>
         This coding is how Value Code Lists and Measured Values are declared.
        <xs:element name="[Name]"  type="[EIPSBasicFormat]Type" minOccurs="0">
```

```
            : (as explained above)
            </xs:element>
            This coding is how simple non-reused Values are declared.
            : Repeat as necessary
            : {Coding for characteristics is defined in section 4.3}
            </xs:sequence>
<xs:attribute name="schemaVersion" type="xs:token"/>
</xs:complexType>
```

Note: Each element is a data object coded as specified in section 4.3.

Name	**Meaning**
EIPSName	Preferred name for the EIPS.
Name	Preferred name for the data item.

Table 4-8 Minimum Information Needed for an EIPS complexType Other Than the Common Content

5. Content Coding Rules

This section provides the following information in the following sequence:

1. Rules applying to all items
2. Rules applying to the common data items
3. Rules applying to Basic Data Format objects
4. Rules applying to Characteristic objects
5. Rules applying to EIPS/XSD objects

5.1. Rules Applying to All EIPS/XSD Items

5.1.1. Spelling

Rule 5-1: Spell check SHOULD always be run on the name and definition fields. Webster MUST be used for the spelling standard. American spelling variants MUST be used instead of British (i.e., color instead of colour).

Rationale: As a community-agreed specification, its content should be expected to be void of spelling errors.

5.1.2. Grammar and Punctuation

Rule 5-2: As a community-agreed specification, its contents SHOULD be expected to contain names and definitions that use legal grammar (within the limits stated above) that conveys clear and unambiguous information about the items.

Rationale: As a community-agreed specification, its content should be expected to be void of grammatical errors.

5.1.3. Code Persistency

Rule 5-3: As long as a MajorRevision change does not dramatically alter the semantics of an item, the Code MUST remain the same.

Rationale: This is essential to support the scenario where the supplier has a different version of the Characteristic or EIPS than the customer, yet still believes the information to be exchanged will be meaningful to the customer. The commonality of the code, in spite of MajorRevision differences, will allow the supplier to do this. However, if the changes made to an item are significant enough such that confusion would result in this scenario, then an entirely new item should be created instead, with a different code.

5.1.4. Delimiters

Rule 5-4: Tag content MUST NOT be enclosed in single or double quotes, parentheses, or any other delimiters, except for a trailing period or other appropriate terminal punctuation in the case of a definition or remark.

Rationale: The starting and terminating tags are the expected delimiters. Any other delimiter just reduplicates the XML delimiters.

Example:

CORRECT

<urss:ReferenceSource>IEC60404-1, JIS C 2502: 1998</urss:ReferenceSource>

INCORRECT

<urss:ReferenceSource>"IEC60404-1, JIS C 2502: 1998"</urss:ReferenceSource>

5.1.5. Trailing Punctuation

Rule 5-5: The content of a ReferenceSource, or other element, whose semantics do not demand a complete or partial sentence, MUST not be terminated with a punctuation mark (such as a period) unless that character is an integral part of the value itself.

Rationale: Punctuation marks that have no semantic meaning waste space and confuse the reader.

Example:

CORRECT

<ReferenceSource>IEC60404-1, JIS C 2502: 1998</ReferenceSource>

INCORRECT

<ReferenceSource>IEC60404-1, JIS C 2502: 1998.</ReferenceSource>

5.1.6. Normative Information versus Non-Normative Information

Some information provided in the appinfo common section is necessary for complete understanding of exactly what the data object is that is being referred to in the XSD structure and thus is necessary for correct mapping to back-end systems. This information sets the norms for the mapping and thus is called normative. Normative information includes short and extended definitions and alphanumeric symbols.

The remaining information facilitates the processing of the information but is not required for correct mapping. This information is called non-normative. It may aid in the mapping of the object but does not set the norm for the mapping. This information includes versioning information and formatting information.

5.1.7. Versioning

Versioning information consists of a major revision number and a minor revision number. When a technical data object is created the major revision number is 1 and the minor revision number is 0.

5.1.7.1. MajorRevision

Rule 5-6: The MajorRevision number of an item MUST be incremented by one and the MinorRevision MUST be set to zero in the case where changes are made to any part of the containing item that cannot be guaranteed to be backward compatible.

Rationale: Users must be warned exactly which items have undergone a change that might affect the back-end mapping of the user.

Rule 5-7: For certain kinds of changes, the MajorRevision MUST always be incremented (refer to the following section 5.1.7.2). But other kinds of changes will have to be evaluated on a case by case basis to determine if the particular change would cause a compatibility issue.

Rationale: Since the user must evaluate whether a major revision will affect back-end mapping, major revisions should only be specified if they are truly necessary.

5.1.7.2. Changes Causing Mandatory Increment

Rule 5-8: The version number MUST be changed if the Name changes (including minor spelling changes).

Rationale: Some solution providers may use the Name as the mapping key. Hence if the spelling changes the mapping must also change.

5.1.7.3. MinorRevision

Rule 5-9: The MinorRevision of an item MUST be incremented by 1 and the MajorRevision MUST NOT be changed in the case where changes to the containing item are backward compatible.

Rationale: This rule will keep to a minimum the items that the user must check for back-end mapping issues.

Examples of MinorRevision changes include spelling, grammatical, or other changes to normative elements that do not affect the meaning, as well as changes to, or additions or deletions of the non-normative tags. The following tags are non-normative in the EIPS/XSD.

- ReferenceSource
- Remark
- ValueFormat

5.1.7.4. Changes Requiring Evaluation

Rule 5-10: Changes that are not simple typographical corrections to a normative element of any item MUST be evaluated to determine if the semantics are affected to the degree where backward compatibility would be compromised, and if so, a MajorRevision increment MUST be required.

Rationale: Any change that potentially compromises back-end mapping must be expressed as a major revision.

5.1.7.5. Versioning of the EIPS/XSD

Rule 5-11: If the content of an EIPS/XSD has been changed, the version of the EIPS/XSD MUST also be changed. The MajorRevision of the EIPS/XSD MUST be incremented by one and the MinorRevision MUST be set to zero if either or both of the following conditions are met:

- If any characteristics were added to or deleted from the EIPS/XSD
- If the changes to the EIPS/XSD resulted in a MajorRevision for one or more of its characteristics

In the case of all other changes, the MinorRevision of the EIPS/XSD MUST be incremented by 1 and the MajorRevision MUST not be changed.

Rationale: The versioning of the EIPS provides the user with an overview of the kinds of changes to its content. If the EIPS only shows a minor revision at the EIPS level, the user need not check further as to whether back-end mapping will be affected. However, if the EIPS shows a major revision, then the user must look at each data item in the EIPS to see whether back-end mapping has actually been affected (e.g, the user may not be using the data item that had a major revision).

5.2. Rules Applying to the Common Data Items

Table 5.1 lists the data fields that are common to the EIPS/XSD, its constituent Characteristics, and the Basic Data Formats used by those Characteristics.

Name	**Meaning**	**Cardinality**	**Example**
Name	Preferred name for the item	1	AluminumElectrolyticCapacitorsWithSolidElectrolyteType
Code	A code that uniquely identifies this item within the namespace to which it belongs	0..1	EIM003
Definition	A combination of the ShortDefinition + ExtendedDefinition and Remark if they exist	1	The community-agreed subset of characteristics for Aluminum Electrolytic Capacitors with Solid Electrolyte. This property set is based in part on ECALS class XJA046.
DateCreated	Date on which this item was created.	1	2004-04-26
DateOfLastMajorRevision	Date that this item last had a major revision	1	2005-07-21
DateOfLastMinorRevision	Date that this item last had a minor revision	0..1	2005-07-21
TypeVersion	A 2-digit MajorRevisionNumber followed by a period followed by a 2-digit MinorRevisionNumber	1	01.00
Alphanumeric Symbol	The preferred symbol represented in 8-bit ascii code	0..1	V_CC
ReferenceSource	An identification of the source of the name and definition if it is obtained from another source, such as the IEC dictionary or the ECALS dictionary	0..n	JIS C 5101-1:1998
Graphic	A line drawing that describes the item	0..n	http://www.rosettanet.org/eim/eips/graphic/capacitor.jpg
ValueFormat	The format of the item	0..1	M..32
Unit	The “Units of Measure” for a measured data item	0..1	

Table 5.1 Data Fields That are Common to the EIPS/XSD, Its Constituent Characteristics, and the Basic Data Formats Used by Those Characteristics

5.2.1. Rules for Name

Rule 5-12: The Name SHOULD match the most commonly used American English name within the industry for which the represented product is used.

Rationale: As a community-agreed item, the name should be the one that is most commonly used in the community.

Rule 5-13: "Upper-Case Camel" MUST be used; that is, the first letter of each word must be capitalized and the rest be lower case, unless the word is an acronym, where it would be all capitals.

Rationale: Current RosettaNet practice.

Rule 5-14: The Name MUST contains only alphanumeric characters. It MUST not contain any spaces or special characters.

Rationale: W3C practice and rules.

5.2.2. Rules for Code

A Code is a coded identifier for the containing item.

Rule 5-15: The content combined with the content of the MajorRevision MUST form a unique key in the namespace of the containing file.

Rationale: Most existing back-end Product Data Management systems use the code as the access key to the back-end system, not the Name.

Rule 5-16: The Code SHOULD, if possible, consist of three or four alphabetic characters followed by three or four digits.

Rationale: Three letters followed by three numbers is standard practice for all technical dictionary codes, but not all RNTD codes follow that rule.

Example: XJG931

5.2.3. Rules for Definition

Rule 5-17: The Definition field MUST contains at least a ShortDefinition. It MAY also contains ExtendedDefinition and Remark.

Rationale: A ShortDefinition is the minimum amount of information needed to recognize the semantics of a Name. However, some semantics for some properties may be quite complex and need additional information to provide complete understanding.

Rule 5-18: The definition MUST not be a simple copy of the item's name.

Rationale: A simple copy of the item's name adds no new information and wastes space, and an effort should be made to avoid reuse of the name at all in the definition.

Rule 5-19: All definition words MUST be spelled correctly. The combination of the short and extended definitions MUST provide a clear and unambiguous meaning for the item. Both of these definitions are normative. Remarks provide non-normative but often useful information.

Rationale: As previously explained in Rule 5-1 and Rule 5-2, an EIPS, as a community-agreed specification is expected to be void of grammatical and spelling errors.

Example**:**

Name: BendingRadius

ShortDefinition: minimum and typical value of the allowed radius measured to the centre-line of the conductor when bending it into a planar curve

ExtendedDefinition: Applying a smaller bending radius than the minimum value may result in damaging the conductor, affecting its function or shorten its lifetime.

Remark: The minimum value applies to a single bending of a conductor and the typical value to the multiple bending of a conductor. The minimum bending radius is often implemented using a template.

“TubeType” would be: “The code of the category to which an electron tube belongs.”

Rule 5-20: The content of ShortDefinition MUST be one phrase free of any XML markup, that is, free-form text.

Rationale: This rule facilitates the use of the ShortDefinition in a mouse-over process.

Rule 5-21: The phrase MUST not repeat the name of the item.

Rationale: To repeat the name of the item in a definition wastes space and may result in a circular definition.

Rule 5-22: The semantics of the content MUST not contradict those of the ExtendedDefinition.

Rationale: Users will be confounded if they encounter two definitions with different meanings for the same property.

5.2.3.1. Rules for ExtendedDefinition

An ExtendedDefinition is an extended version of the normative definition of the item.

Rule 5-23: The semantics of the content MUST not contradict those of the ShortDefinition.

Rationale: As previously explained in Rule 5-22 immediately above, users will be confounded if they encounter two definitions with different meanings for the same property.

Rule 5-24: The additional, normative details included in the ExtendedDefinition MUST all be complete, grammatically correct sentences.

Rationale: As previously explained in Rule 5-2, an EIPS, as a community-agreed specification is expected to be void of grammatical errors.

5.2.3.2. Rules for Remark

Rule 5-25: Comments, notes, or other non-normative remarks about the item MAY be provided.

Rationale: If the 15 year history of technical dictionaries, it has proven to be useful to occasionally provide non-normative information such as how to know when a certain property should be provided with another property.

5.2.4. Rules for MajorRevisionNumber

Rule 5-26: This number MUST be a two digit number with initial zeros. It is initially “01” and MUST be incremented by 1 each time this item is revised in such a way as to affect mappings using this item.

Rationale: The two digit number rule is current RosettaNet practice and the increment rule reflects the versioning rules in section 5.1.7.

5.2.5. Rules for MinorRevisionNumber

Rule 5-27: This number is a two digit number with initial zeros. It is initially “00” and MUST be incremented by 1 each time this item is revised in a way that does not affect mappings using this item; and MUST be reset to”00” when a major revision occurs.

Rationale: The two digit number rule is current RosettaNet practice and the increment rule reflects the versioning rules in section 5.1.7.

5.2.6. Rules for CreationDate

Rule 5-28: This field MUST contain the date that this item was initially defined.

The content MUST be a 10-character date in IS0 8601 format, e.g., CCYY-MM-DD: "1994-03-21".

Rationale: This is current RosettaNet practice.

5.2.7. Rules for LastUpdatedDate

Rule 5-29: This field MUST contain the date that this item was last changed. This is the latest date of either the DateOfMajorRevision or DateOfMinorRevision.

Rationale: To be consistent with RosettaNet practice, which does not distinguish the dates of major and minor updates.

Rule 5-30: The content MUST be a 10-character date in IS0 8601 format, e.g., CCYY-MM-DD: "1994-03-21".

Rationale: This is current RosettaNet practice.

5.2.8. Rules for TypeVersion

Rule 5-31: This field MUST contain a 5-digit code consisting of the last two digits of the MajorRevisionNumber, followed by a period, followed by the last two digits of the MinorRevisionNumber.

Example: 01.03

Rationale: This is current RosettaNet practice.

5.2.9. Rules for AlphanumericSymbol

Rule 5-32: This field MAY contain an abbreviated, plain text representation of the PreferredSymbol, MUST utilize Latin letters only, and MUST be constructed according to the following rules:

1. The first character MUST be a letter, "@", or a "$"when defining a specific Greek letter. If applicable, the "@" sign precedes the "$" sign.
2. The length MUST be limited to 15 characters.
3. Greek letters that are graphically identical to Latin letters are not converted.
4. 0ther Greek letters MUST be represented by the corresponding ISO/R843 Latin letter preceded by a dollar "$" character.
5. An asterisk "*" character MUST be used to indicate that the following letter is a superscript.
6. An underline "_" character MUST be used to indicate that the following letter is a subscript.

Rationale: The symbols used to identify technical properties often use subscripts, superscripts, and Greek letters. Many of these, of course, cannot be expressed in ascii (Latin) characters so they have to be converted by some standard rule.

5.2.10. Rules for ReferenceSource

Rule 5-33: This field MAY contain a non-normative reference to the document from which a significant portion of the semantics of the parent element was derived. (Note: No standard encoding has been defined to make the content machine-sensible.)

Rationale: This reference is required by our MOUs with IEC and ECALS. It also shows that we respect the intellectual property of the sources.

Rule 5-34: There MUST be a ReferenceSource for each distinct reference.

Rationale: If the definition was taken from IEC or ECALS, which in turn took the definition form some other ISO or IEC standard, then both must be referenced to protect the intellectual property of both parties.

5.2.11. Rules for Graphic

Rule 5-35: This tag MAY point to a non-text object. The pointer MUST be coded in such a way the person viewing the XSD will be able to see the graphic with the most common XML viewing and editing software.

Rationale: Some technical properties define a complex curve that will be easier to understand if shown in a graphic rather than trying to explain curves and line angles in text alone.

Example:

<urss:AlphanumericSymbol>t_rr</alphanumeric.symbol>

<urss:Definition>The time interval between the instant when the current passes through zero when changing from the forward direction to the reverse direction and, after reverse current reaches its peak value IRM(REC), the instant when the extrapolated reverse current reaches zero, as shown in the attached Figure. The extrapolation is a straight line drawn through the reverse recovery current as it reduces from 90% of peak reverse recovery current IRM(REC) to 50% of IRM(REC).</urss:Definition>

<urss:Graphic>http://www.rosettanet.org/eim/eips/graphic/capacitor.jpg</urss:Graphic>

5.3. Rules Applying to Basic Data Format Objects

As explained in section 4.2, the Basic Data Format objects will have the following additional fields.

Name	**Meaning**	**Cardinality**
ValueFormat	Format in human-readable form based on IEC 61360, selected from the Value column of Table 6.2	1
Pattern	Selected from the Validation General Expression column of Table 6.2 in accordance with the above ValueFormat	1

Table 5.2 Additional fields in a Basic Data Format item

Rule 5-36: The appropriate ValueFormat MUST be selected from the “Value” column of Table 5.3.

Rationale: As previously explained in Rule 4.8, if the correct ValueFormat is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.

Rule 5-37: The pattern MUST be expressed within Validation General Expression in Table 5.3 that corresponds to the Value defined in ValueFormat.

Rationale: As previously explained in Rule 4.8, if the correct general expression pattern is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.

Value	Code	Name	Validation General expression	Comment	Reused Structure
B 1	--	Boolean	--	XML Data Type	Boolean
M..10	EBF001	EIPSString10BasicType	maxLength value="10”	Restriction	String
M..20	EBF002	EIPSString20BasicType	maxLength value="20”	Restriction	String
M..32	EBF003	EIPSString32BasicType	maxLength value="32”	Restriction	String
M..60	EBF004	EIPSString60BasicType	maxLength value="60”	Restriction	String
M..100	EBF005	EIPSString100BasicType	maxLength value="100”	Restriction	String
M..240	EBF006	EIPSString240BasicType	maxLength value="240”	Restriction	String
M..255	EBF007	EIPSString255BasicType	maxLength value="255”	Restriction	String
M..2048	EBF008	EIPSString2048BasicType	maxLength value="2048”	Restriction	String
N 9	--	DUNSType	--	Universal Data Type	DUNSType
N 4	EBF009	EIPSNumeric4BasicType	¥d{4}		Integer
N..10	EBF010	EIPSNumeric10BasicType	¥d{1,10}	Restriction	Integer
N 14	--	DUNSPlus4Type	--	Universal Data Type	DUNSPlus4Type
NR1 S..10	EBF011	EIPSSignedInteger10BasicType	[¥+¥-]?¥d{1,10}	Restriction	Integer
NR2..2.3	EBF012	EIPSUnsignedReal5BasicType	¥d{1,2}([¥.]¥d{0,3})	Restriction	Real
NR3 S..3.3ES2	EBF013	EIPSSignedFloat6BasicType	[¥+¥-]?¥d{1,3}([¥.]¥d{0,3})[E][¥+¥-]?¥d{1,2}	Restriction	Float
NR3 S..7.7ES2	EBF014	EIPSSignedFloat14BasicType	[¥+¥-]?¥d{1,7}([¥.]¥d{0,7})[E][¥+¥-]?¥d{1,2}	Restriction	Float
NR3..3.3ES2	EBF015	EIPSUnsignedFloat6BasicType	¥d{1,3}([¥.]¥d{0,3})[E][¥+¥-]?¥d{1,2}	Restriction	Float
X 16	--	Date	--	XML Data Type	Date

Table 5.3 Value Format Values, Names, and Validation Expressions

5.4. Rules Applying to Characteristic Objects

Rule 5-38: Depending on the type of data the Characteristic is defining, some of the following additional fields MAY appear.

Rationale: The lower left half of the data model in chapter 3 shows that measured values may consist of either a simple value or a set of three (toleranced) values or up to four min/max values.

Name	**Meaning**	**Cardinality**
NameOfBasicDataFormat	Name selected from the name column of Table 5.3 and inserted wherever the dataType must be encoded.	1
Unit	The Unit Of Measure applied to this value. This item, if required, will be under appinfo.	0..1
ValueFormat	Description of format using IEC 61360-style format selected from the value column of Table 5.3. This item will be under appinfo.	1

Table 5.4 Additional Information Applying to Characteristic Objects

5.4.1. NameOfBasicDataFormat

Rule 5-39: The NameOfBasicDataFormat MUST be a name in Table 5.3 that corresponds to the contents of the ValueFormat in the same Characteristic.

Rationale: If the above rule is not followed the property will not validate correctly.

5.4.2. ValueFormat

Rule 5-40: The ValueFormat MUST be a value in the value column of Table 5.3 that will cause the appropriate validation of the content of the Characteristic.

Rationale: If the above rule is not followed the property will not validate correctly.

5.4.3. ValueCode

Rule 5-41: The instance of a code in the message MUST be one of the ValueCodes provided in the value code list of this Characteristic.

Rationale: The value code list defines the domain of acceptable values. Anything outside of this domain is an error.

Rule 5-42: The ValueCode MUST not duplicates another ValueCode in the same list.

Rationale: Allowing duplicate values will confuse the computer when validating the message.

5.4.4. ValueCodeMeaning

Rule 5-43: This field MUST contain a human-understandable explanation of the meaning of the ValueCode.

Rationale: The whole point of providing the meaning is to allow a human to understand the meaning of the value code.

Example:

If the ValueCode was “SRAM,” the ValueCodeMeaning should be “static random access memory.”

5.4.5. Unit

Rule 5-44: The content of the Unit field SHOULD be as specified within one of the following standards for unit of measure.

- International System of Units (SI) (http://physics.nist.gov/cuu/Units/index.html)
- IEC 61360-1 (Annex B) (http://www.iec.ch/)

Rationale: Both are internationally agreed lists of units of measure and cover 99.8 percent of units of measure needed for electronic components.

Note: The rules in IEC 61360-1 are a small extension of the SI units, especially applicable to electronic components.

Rule 5-45: The units of measure MAY also include the following additions to the above standards for units of measure: ppm (parts per million) and % (percent).

Rationale: Both of those additions are needed for technical specifications but are not part of either of the above standards.

5.4.5.1. Unit of Measure Encoding When SI Unit is Appropriate

Rule 5-46: When an appropriate SI Unit exists for the property to be defined, the Units of Measure MUST be defined using the content of the UOM tables in Section 3.2.1 (Constructing Units of Measure) of the EIPS/XSD Creation/Maintenance guideline.

Rationale: The tables in the *EIPS/XSD Creation/Maintenance guideline* have been extracted from the standards required by rule 5-43.

5.4.5.2. Unit of Measure Encoding When No SI Unit Applies

Rule 5-47: Units not covered by Section 3.2.1 (Constructing Units of Measure) of the *EIPS/XSD Creation/Maintenance guideline* MUST have a “1” coded in the Unit field, and the item’s short definition, and optionally its name, MUST specify the meaning of the measurement.

Rationale: International standards typically cover all scientific units of measure but not all engineering units of measure. Take speed for an example: there is an SI unit for physical distance covered over time but no SI unit to explain the speed of a CPU (which does not move over some physical distance but rather is computing a number of instructions over time). These engineering units of measure are ignored by ISO and IEC but we need to define them. In that case, there is no ISO or IEC unit to be coded in the unit field, so the unit is replaced with a “1”.

Examples of measurements not covered in Section 3.2.1:

Characteristic Name	EIPS Encoding	Characteristic Definition
Number of pads	**1**	The number of electrical pads on the component.
Execution Speed	**1**	Instructions per second

Table 5.5: Characteristics that Do Not Have a SI Unit.

5.4.6. Units of Measure Consistent with Definition

Rule 5-48: The unit of measure for an item MUST be consistent with the name and definition of that item. For example, if the name or definition refers to a voltage or current value, then the unit of measure MUST be in volts or amperes (respectively).

Rationale: A community-agreed standard should not confuse users with units of measure that contradict the name or definition of the Characteristic.

Examples of illegal items are:

Name	Unit	Error
Cycle Time	hertz	**Time** in the name infers **seconds** for the unit of measure
Number Of Bolts	meter	**Number** in the name infers units other than **length**
Head Height	1	**Height** in the name infers a unit of measure of **length**, thus it will among the SI units
Acquisition Time	bit	**Time** is not measured in **bits**

Table 5.6 Examples of Units of Measure that Conflict with a Characteristic Name

5.4.7. Units Encoding Requirements

Rule 5-49: Units that consist solely of alphanumeric characters, including characters that can be represented as XML character entities, MAY be entered literally. However, if the symbol consists of any math operation (such as explicit or implied multiplication, division, or exponentiation, etc.), or any display format information (such as subscripting, superscription, overbar, prime, etc.), then the symbol must be encoded in MathML (Presentation style).

Rationale: Units of measure are often expressed in mathematical expressions. MathML is a well-known W3C standard for expressing mathematical expressions.

Examples:

Equation Format	Description	MathML Representation
x+y	x plus y	<math> <mi>x</mi><mo>+</mo><mi>y</mi> </math>
Second		***-- NONE REQUIRED --***
mx + b	M times x, plus b	<math><mrow><mi>m</mi> <mo>&InvisibleTimes;</mo><mi>x</mi> </mrow><mo>+</mo><mi>b</mi> </math>
x**3	x cubed	<math><msup><mi>x</mi><mn>3</mn></msup></math>
1/second	per second	<math><mn>1</mn><mo>/</mo><mi>second</mi></math>
y**2/(joule-second)	Y squared, per joule second	<math> <mrow> <msup><mi>y</mi><mn>2</mn></msup> </mrow> <mo>/</mo><mi>(joule-second)</mi> </math>

Table 5.7 Examples of MathML Representation

5.5. Rules applying to EIPS/XSD objects

Rule 5-50: The Characteristics in the EIPS/XSD MUST be coded in alphabetical order by Name.

Rationale: This is current RosettaNet practice.

6. Versioning Rules

6.1. EIPS/XSD Versioning and Releases

Versioning information consists of a major revision number and a minor revision number. A major revision is defined as a change that MAY force current users to make a change in their coding or mapping tables. A minor revision is defined as a change that does not force current users to make a change in their coding or mapping tables.

Rule 6-1: When a technical data object is created the major revision number is 01 and the minor revision number is 00.

Rationale: This is current RosettaNet practice.

6.2. Basic Rules

Rule 6-2: If a change to EIPS/XSD content MAY force current users to make changes in their coding or tables, then the change MUST be a major revision. Otherwise the change MUST be a minor revision.

Rationale: As explained in the Versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.

Examples of MinorRevision changes include spelling, grammatical, or other changes to normative elements that do not affect the meaning, as well as changes to, or additions or deletions of the non-normative tags. The following tags are non-normative in the EIPS/XSD.

- ReferenceSource
- Remark
- ValueFormat

Rule 6-3: If the change does not change any current functionality but enables new functionality and is optional, it will not affect current coding or tables unless the user wants that new functionality. This kind of change MUST be a minor revision.

Rationale: As explained in the Versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.

For example, adding a new CharacteristicDefinition, a new ContentType, or even a new class just adds functionality but does not change current functionality. On the other hand changing a current class, PropertyDefinitionSet, or CharacteristicDefinition in an essential way MAY change current processing and thus is a major revision.

Rule 6-4: Version numbers MUST be two digits long with leading zeroes.

Rationale: This is current RosettaNet practice.

As specified in Rule 6-1, all new items have a MajorRevisionNumber of 01 and a MinorRevisionNumber of 00.

Rule 6-5: For minor revisions of items, the MinorRevisionNumber is increased by 1. For major revisions, the MajorRevisionNumber is increased by 1 and the MinorRevisionNumber is reset to 00.

Rationale: This is current RosettaNet practice.

6.3. Detailed Versioning Rules for CharacteristicDefinitions in an EIPS

Rule 6-6: A major revision MUST occur when any change is made to the Name.

Rationale: Some solutions MAY use the Name as the mandatory access point to the EIPS.

Rule 6-7: A major revision MUST occur when non-editorial changes are made to the ID, definition, or domain ("type," Units, ValueFormat, or ValueDomain).

Rationale: Many solutions use the ID as the access point to the EIPS and to the backend system. Non-editorial changes to the definition MAY change the scope of its semantics and thus the mapping to the backend system. Changes to the domain MAY require changes to subroutines that convert the data in the message instance before it is sent to the backend system.

Rule 6-8: Obsolescing MUST result in a major revision.

Rationale: If a user is using this item, then the user’s coding and/or tables must be revised.

Rule 6-9: All other changes MUST be minor revisions.

Note: The definition is said to change not only if its meaning changes but also if its scope of application changes; that is, if a property is the merge of two or more existing properties or the bifurcation of a single property into two or more properties. E.g., ECALS has split Supply Voltage into Single Supply Voltage and Dual Supply Voltage for some classes. This change results in a major revision.

Rationale: Users who previously used the property called Supply Voltage in the classes where the scope of meaning was divided between single supply voltage and dual supply voltage will have to recode their mapping to include either or both of these new properties with their new codes. However, users of the same property in classes where the original concept still applies can continue to use the old property and need not update their mappings.

6.4. Detailed Versioning Rules for EIPS Definition Itself

Rule 6-10: A major revision MUST occur when any change is made to the Name.

Rationale: Some solutions MAY use the Name as the mandatory access point to the EIPS.

Rule 6-11: A major revision MUST occur when non-editorial changes are made to the ID or Definition. The Definition is said to change if its scope of application changes.

Rationale: Many solutions use the ID as the access point to the EIPS and to the backend system. Non-editorial changes to the definition MAY change the scope of its semantics and thus the mapping to the backend system.

Rule 6-12: Obsoletion MUST result in a major revision.

Rationale: If a user is using this item, then the user’s coding and/or tables must be revised.

Rule 6-13: A major revision MUST occur when a change is made in the sequence of Characteristics currently defined in the EIPS. Adding Characteristics to the end of the EIPS is a minor revision.

Rationale: This rule is required because the sequence of Characteristics is defined as an "ordered" list of CharacteristicDefinitions, and solution providers may have coded a dependence on the ordering. Since the items are all optional (but ordered), additions at the end should not affect current coding.

Rule 6-14: All other changes MUST be minor revisions.

Rationale: As explained in the versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.

6.5. EIPS File Versions and Releases

Rule 6-15: If any item in a given EIPS file undergoes a major revision during a maintenance cycle, the next release of that EIPS file MUST be a major revision.

Rationale: As explained in the Versioning rules in section 5.1.7, users must be warned that some items have undergone a change which might affect the back-end mapping of the user.

Rule 6-16: If the file undergoes a major revision, the versioning of the file name MUST follow RosettaNet rules for the versioning of files.

Rationale: As explained in the Versioning rules in section 5.1.7, versioning of technical information items should follow RosettaNet rules to minimize errors.

Rule 6-17: Obsolete items MUST be kept in the released EIPS file for one release cycle and then not output in any later releases.

Rationale: Although an item has been obsolesced, the user will have one maintenance cycle within which to update the mapping tables to the new version of the EIPS. In the meantime the old item can still be used if both partners agree.

Rule 6-18: Items to be obsoleted MUST be described in the accompanying release documentation.

Rationale: Although an item has been obsolesced, the user will have one maintenance cycle within which to update the mapping tables to the new version of the EIPS. In the meantime the old item can still be used if both partners agree.

Note: Maintenance cycles and release cycles can be independent, that is a maintenance cycle for a given set of EIPS items can be shorter or longer than the release cycle. If all ongoing maintenance work cannot be completed before a scheduled release date (minus the prescribed time for release QA and authorization process), the ongoing maintenance work will be stopped long enough to offload all validated, un-obsolete items into a separate file or workspace for release QA and authorization processing. Once the latest version is offloaded, maintenance processing may continue.

7. EIPS Packaging

7.1. Interrelation of Data Modules Under a TPIR PIP with Engineering Information

The TPIR PIP with Engineering Information is highly modularized. The following diagram shows the high degree of modularity.

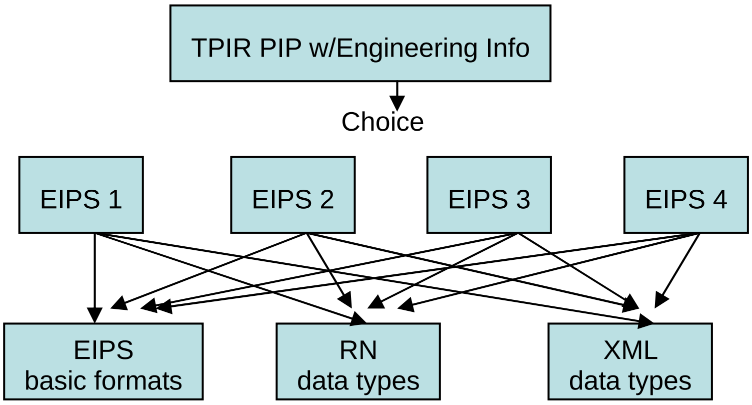


Figure 7.1 Modularity in TPIR PIPs with Engineering Information.

7.2. EIPS/XSD File, Namespace, Namespace Prefix, and Versioning Guidelines

Rule 7-1: The EIPS/XSD file name MUST comply with the following requirements:

Requirement 7-1-1: The EIPS/XSD applies to MUST be easily identified.

Requirement 7-1-2: The EIPS/XSD file name MUST identify its versioning (major and minor version numbers).

Rationale: Compliance with RosettaNet practices.

Example: The standard format of the EIPS/XSD file name is "[EIPS_Name]_[EIPSMajorVersion_EIPSMinorVersion].xsd"

So if EIPS is describing an product class and the name of the product class is "solid tantalum fixed capacitor," then the file name typically would be:

CapacitorFixedTantalumSolid_01_00.xsd

Note: the product class name is written in reverse order to enable similar product classes to be grouped together. This rule for reversing the order of a product class name only applies to an EIPS that is describing a product class. It does not apply to other data objects.

The rules for versioning are explained in sections 5.1.7 and 6.2.

Rule 7-2: The EIPS/XSD MUST comply with TPIR-PIP namespace and namespace prefix rules.

Rationale: Since the TPIR-EIPS is created based upon a TPIR-PIP, the TPIR-EIPS namespace and namespace prefix must comply with TPIR-PIP rules.

Rule 7-3: The subtype field in the EIPS/XSD namespace specific string MUST be “eips.”

Rationale: By setting the subtype to eips, the application can deduct that the file contains an EIPS/XSD.

Example: When Capacitor-Fixed-Tantalum-Solid_01_00.xsd is used, the namespace appears as follows: xmlns:ecfts="urn:rosettanet:specification:domain:Design:
Capacitor-Fixed-Tantalum-Solid:xsd:eips:01.00" where “ecfts” identifies the namespace prefix.

Rule 7-4: The EIPS/XSD namespace prefix MUST start with an "e" for EIPS then followed by the initial letter of every word in the name. E.g., If the name is Capacitor-Fixed-Tantalum-Solid, the the namespace prefix would be “ecfts”. If the prefix duplicates another existing prefix, then the duplication is eliminated by adding a suffix letter, starting with “a” and continuing through the alphabet until no duplication exists.

Rationale: The namespace prefix should be generated using simple rules and should be persistent for a specific EIPS.

Example: If “ecfts” is already being used by a product class, the new prefix should be “ecftsa”. If “ecftsa” is already being used, the new prefix should be “ecftsb”.

7.3. EIPS File Structure and Contents

Rule 7-5: Each EIPS MUST be contained in its own separate file.

Rationale: EIM program requirement. It also reinforces modularity for all TPIR PIPs with Engineering Information.

Rule 7-6: The file MUST contain all element declarations in alphabetic order, and all the EIPS complexType definitions containing all the element declarations in alphabetic order.

Rationale: This structure and sequencing is standard RosettaNet practice.

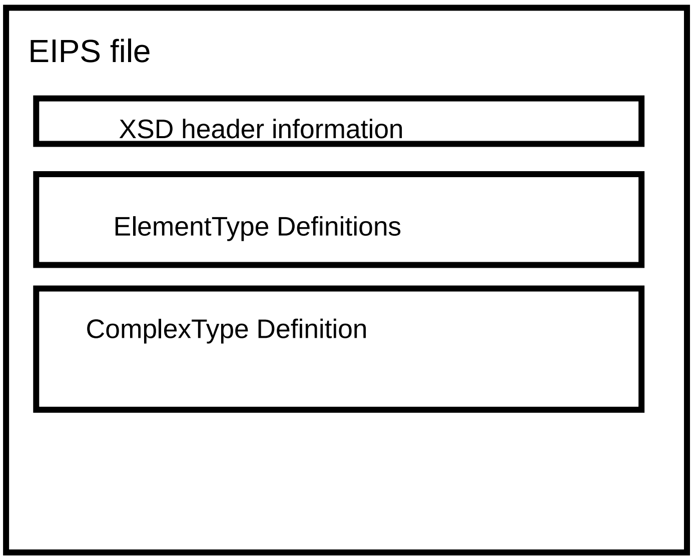


Figure 7.2 Structure and Sequencing of EIPS File Contents

7.3.1. XSD Header Information

Rule 7-7: Each EIPS MUST define its own namespace. The namespace prefix must be persistent.

Rationale: RosettaNet utilizes a homogenous Namespace concept. The persistence of a namespace prefix simplifies implementation of multiple EIPS by implementers. It also works as a unique identifier for XML Schema files inside RosettaNet.

7.3.2. EIPS Element Type Definitions

This section contains the type definitions for value code lists and measured values in alphabetic order.

7.3.3. EIPS complexType Definition Containing Data Element Declarations

This section contains the EIPS complexType definition and the data elements it contains, declared in alphabetic order. The definitions for all non-reused simple data types are here.

7.4. EIPS Delivery Package Structure and Contents

7.4.1. Contents Overview

Rule 7-8: Each of the EIPS delivery packages MUST have the following structure and content.

Rationale: This structure complies with the structure and contents of the latest RosettaNet PIP delivery packages, ignoring items, such as the BPSS, that do no apply to an EIPS.

```
[EIPS folder]
    [ReadMe.txt file]: explanation of the contents of the EIPS folder, including disclaimers.
    [Descriptive folder]: contains all non-normative files
        [EIPS.csv file]
        [EIPS.xml message coding sample file]
        [Release Notes]
    [XML folder]: contains all normative files
        [Domain folder]
            [Design folder]
                [EIPS.xsd file]
                [EIPSBasicFormat.xsd file]
                [Code Lists folder]
            [Universal]
                [CodeList]
                    [ISO_Language.xsd file]
                [ContactInformation.xsd file]
                [DataType.xsd file]
                ….
```

The ReadMe.txt file is a revised version of the standard PIP readme file that correctly describes the structure and contents of an EIPS delivery package.

The EIPS.csv file is a spreadsheet that shows the structure of the EIPS.xsd file.

The EIPS.xml file is a generalized example of how an EIPS message segment (each EIPS.xsd defines a message segment) would look.

The Release Notes describe the purpose of the EIPS and any modifications that have occurred since the last release.

The EIPS.xsd file is the normative version of the EIPS.

The EIPSBasicFormat.xsd file is the normative version of most of the data formats used in the EIPS.

Rule 7-9: The EIPS/XSD file MUST be stored under the Design folder.

Rationale: These definitions are only used in the domain of design engineering and are not used in any other domain.

Rule 7-10: The EIPSBasicFormat.xsd MUST be included with the EIPS.xsd file.

Rationale: The use of the basic format file increases the modularity of TPIR PIPs with Engineering Information and ensures that each EIPS has the correct version of the basic format file.

8. References

Source	Description
EIM Engineering Report	Title : EIM Investigation Report V01.00 (15June2005) RosettaNet Retrieved March 03, 2008 from: http://members.rosettanet.org/dnn_rose/Standards/RosettaNetPrograms/FoundationalPrograms/ActiveFoundationalPrograms/EIMFoundational/InvestigationPhaseMaterials/tabid/1431/Default.aspx
[Guide to Versioning XML Languages using XML Schema]	Title: Guide to Versioning XML Languages using XML Schema 1.1. World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/xmlschema-guide2versioning/
[ISO/R843]	International System for the Transliteration of Greek Characters into Latin Characters
[ISO 8601:2000 Second Edition]	Title: Representations of dates and times, second edition, 2000-12-15 ISO (International Organization for Standardization)
[Mathematical Markup Language]	Title: Mathematical Markup Language (MathML™) 1.01 Specification World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/REC-MathML/
[Namespaces in XML]	Title: Namespaces in XML World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/1999/REC-xml-names-19990114/
[NSM]	Title: RosettaNet Namespace Specification and Management Issue 01.00.00 (11Dec2003) RosettaNet Retrieved March 05, 2008 from: http://members.rosettanet.org/dnn_rose/DMX/tabid/2979/DMXModule/624/Command/Core_ViewDetails/Default.aspx?EntryId=4335
[RFC2119]	Author: Scott Bradner Title: Key words for use in RFCs to Indicate Requirement Levels The Internet Engineering Task Force Retrieved March 05, 2008 from: http://www.ietf.org/rfc/rfc2119.txt

TPIR-PIP for EI	Title: RosettaNet TPIR-PIP for Engineering Information Specification V1.00.00 (05March2008) RosettaNet Retrieved March 05, 2008 from: http://members.rosettanet.org/dnn_rose/DMX/tabid/2979/DMXModule/624/Command/Core_ViewDetails/Default.aspx?EntryId=325
[TPIR-PIP-DG]	Title: RosettaNet Automated Enablement, Trading Partner Implementation Requirements-Partner Interface Process™ (TPIR-PIP) Design Specification – V11.00.00 RosettaNet Retrieved March 05, 2008 from: http://members.rosettanet.org/Standards/RosettaNetStandards/TradingPartnerImplementationRequirements/tabid/480/Default.aspx
[XDG]	Title: RosettaNet XML Design Guideline Issue 01.00.00 (11Dec2003) RosettaNet Retrieved March 05, 2008 from: http://members.rosettanet.org/dnn_rose/DMX/tabid/2979/DMXModule/624/Command/Core_ViewDetails/Default.aspx?EntryId=4335
[XML 1.0 - Second Edition]	Title: Extensible Markup Language (XML) 1.0, Second Edition World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/2000/WD-xml-2e-20000814
[XML Schema Part 1: Structures]	Title: XML Schema Part 1: Structures World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/structures.html
[XML Schema Part 2: Datatypes Second Edition]	Title: XML Schema Part 2: Datatypes Second Edition World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/xmlschema-2/
[XML Schema: Component Designators]	Title: XML Schema: Component Designators World Wide Web Consortium Retrieved March 05, 2008 from: http://www.w3.org/TR/xmlschema-ref/

9. Glossary

Word(s)	Definition
AlphanumericSymbol	The symbol for an item represented in 8-bit ascii code.
Basic Data Type	One of the commonly used formats for technical data.
Characteristic	A feature, parameter, quality, aspect, or property of technical information.
Code	A code that uniquely identifies this item within the namespace to which it belongs.
CreationDate	A field that contains date that this item was first created.
DateOfLastMajorRevision	Date that this item last had a major revision in CCYY-MM-DD format.
DateOfLastMinorRevision	Date that this item last had a minor revision in CCYY-MM-DD format.
Definition	A combination of definitions and remarks to aid in mapping the item.
Dictionary	Any organized set of terms with semantic information about them. American Heritage: "A book listing words or other linguistic items in a particular category or subject with specialized information about them".
EIPS	Engineering Information Property Set. The community-agreed subset of the full set of characteristics for a specific piece of technical information.
EIPS/XSD	EIPS content represented in W3C XML Schema. Used wherever a distinction between the EIPS content in general and its XML Schema rendering is important.
Engineering Information Set	Any set of logically related information that engineers need to perform their job. E.g., product performance data or product manufacturing data.
Format	The syntactical format of a value.
Graphic	A line drawing that describes the item when a textual description is insufficient to fully explain the item.
Item	The coding defining an EIPS or the coding defining a Characteristic in an EIPS.
MajorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in such a way as to affect mappings using this item.
MinorRevisionNumber	A 2 digit number that is incremented by 1 each time this item is revised in a way that does not affect mappings using this item and is reset to 0 when a major revision occurs.
Name	Preferred name for the item.
NameSpace	An abstract container providing context for the items (names, or technical terms, or words) it holds and allows disambiguation of items having the same name (residing in different namespaces).
Non-normative information	Information about a data object that does not specifically define it. It may aid in the mapping of the object but does not set the norm for the mapping. This information includes versioning information and formatting information.
Normative information	Information about a data object that defines the semantics for the

	data object. This information must be clearly understood to correctly map the data object. Hence this information sets the norms for the mapping. Normative information includes short and extended definitions and alphanumeric symbols.
Product class	The generic name for a set of product instances that are defined with the same set of characteristics.
PropertyDefinitionSet	The definition of the semantics and related information of an ordered set of Characteristics.
ReferenceSource	An identification of the source of the name and definition if it is obtained from another source, such as the IEC dictionary or the ECALS dictionary.
Remark	Non-normative additional information in support of mapping, such as why this item should be used for a given concept rather than a similar item.
RNBD	RosettaNet Business Dictionary. The generic name for the set of RosettaNet definitions for business characteristics and their related structures.
Value	The actual data that is carried in a message instance.
Value type	The data type for a given value. See *XML Schema: Data types* for further information on data types. http://www.w3.org/TR/xmlschema-2/

10. Requirements

Requirement 2-1: Every item must be uniquely identified by identifiers.	**Rationale**: Non-unique identifiers can cause parsing errors. The name and/or code (plus major revision) are typically used as the key for backend mapping so both should be kept unique to prevent errors in the business processing side too.
Requirement 2-2: Every item must provide sufficient semantics to aid in mapping data into backend systems.	**Rationale**: Data is provided in a message instance in order to be stored in a back-end system. That data will be useless unless it can be correctly mapped into a given back-end system. The identifiers uniquely identify EIPS/XSD and Characteristics. The Semantics provide information necessary to correctly map the contents of the value to a specific place in a backend database. The values provide information on the value type and its required format in support of error-checking the value.
Requirement 2-3: Every Characteristic MUST contain a value.	**Rationale**: A null value in a message just wastes communication bandwidth.
Requirement 2-4: To support error-checking, each data value included in a message instance MUST be defined by a data type and the format for that data type.	**Rationale**: Technical information databases are a relatively recent development and their content has not been validated as much as the content in business databases, resulting in a noticeable amount of invalid data. Furthermore, only the largest suppliers already have databases in place, whereas most SME(s) keep their technical data either in electronic document or in a paper form and re-enter the data for RosettaNet messages.
Requirement 2-5: The definition for that value MUST also define the units for that value where possible.	**Rationale**: If the dictionary defines the units, conversion between units (such as inches to centimeters) becomes unnecessary and the message need not carry that information.
Requirement 2-6: The EIPS/XSD coding SHOULD match current RosettaNet practice wherever possible.	**Rationale**: Compatibility reduces system development costs.

11. Rules

Rule 4-1: All the data objects in an EIPS/XSD MUST include the tags minimally required by RosettaNet practice.	**Rationale**: Requirement 2.6.
Rule 4-2: All the data objects in an EIPS/XSD MUST include the data items shown in Table 4.1 when their cardinality is 1 or more and MAY include those optional items when the partner's data requirements specify that they are required.	**Rationale**: Requirements 2.1 and 2.2.
Rule 4-3: To satisfy rule 4-1 and 4-2, the XSD coding MUST include the following common items in the appinfo section of the annotation section: urss:CreationDate, urss:Definition, urss:LastUpdatedDate, urss:TypeVersion.	**Rationale**: Although these items are primarily intended for human recognition (to assist in the mapping of the information to a backend system), a solution provider may want to be able to access them for value-added processing, thus they have to be in the appinfo section.
Rule 4-4: The semantic information for a technical data item MUST allow for the optional coding of an AlphanumericSymbol, Graphic, and multiple normative and non-normative Definition content.	**Rationale**: A Definition alone may not be sufficient or efficient for fast mapping to backend information. For many engineers the alphanumeric symbol has greater recognition value while for other Characteristics a graphic may be necessary. At the same time, the concepts of a short definition, extended definition, and remark are standard practice in technical dictionaries (they are defined in IEC 61360) while RosettaNet does not distinguish the concepts. To maintain reasonable compatibility with both practices, the short definition, extended definition, and remark have to be combined.
Rule 4-5: All data items MUST be checked for format and perhaps also for value range.	**Rationale**: Requirement 2-4.
Rule 4-6: The formats MUST be standardized, using an external set of standard formats.	**Rationale**: One of the most important purposes of an EIPS/XSD is to improve error checking of technical content To make the definition of common data formats easier and to minimize coding errors when defining characteristics, an EIPS/XSD uses the concept of "basic data types". The use of these standard formats will make the creation of new EIPS/XSDs easier and also facilitate the definition of the related fields in the back-end database.
Rule 4-7: Each of the basic data types MUST consist of the following tags: xs:simpleType, xs:annotation, xs:appinfo, urss:Code, urss:CreationDate, urss:Definition, urss:LastUpdatedDate, urss:TypeVersion, urss:ValueFormat, xs:restriction.	**Rationale**: Because a Basic Data Format is just a restriction on a real Characteristic, some of the common data elements do not apply and can be deleted from the structure of the Basic Data Format.

Rule 4-8: The content of xs:pattern field MUST be based on the basic type "xs:string" and MUST be the "general expression" pattern in Table 5.3 that corresponds to the format in the ValueFormat field.	**Rationale**: If the correct general expression pattern is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.
Rule 4-9: The standardized structure for simple, unmeasured data types (that are not value codes and that are not reused) MUST consist of the following codes: xs:element, xs:annotation, urss:ValueFormat.	**Rationale**: Simple data elements that are not reused are declared using xs:element according to RosettaNet practice. The xs:annotation tag is necessary to introduce the Common Content, and the urss:ValueFormat is necessary for the human who maps this data to the backend system to understand what the format of the item will be.
Rule 4-10: The xs:element tag MUST define its type attribute as the basic data type listed in Table 5.3 that corresponds to the content of ValueFormat.	**Rationale**: Using one of the basic data types will reduce the possibility of a coding error.
Rule 4-11: The value code list MUST follow RosettaNet design rules for defining code lists, as well as including the common elements. That means that the value code list MUST be specified in a separate file stored in the directory Code Lists under the Design directory. If the EIPS/XSD needs any of these value code lists, they MUST be imported.	**Rationale**: Requirement 2.6
Rule 4-12: The coding for a value that may potentially be any of the three structures MUST be a choice between the three structures, unless the required structure is defined in the definition of the Characteristic.	**Rationale**: Normally, when the EIPS/XSD is being defined, the definers will not be able to determine which of the three structures the partners will choose. However, the definers should look at the definition of the characteristic because sometimes it will state that a given characteristic is expected to be exchanged with a certain set of values, such as min, typ, and max.
Rule 4-13: Because the coding uses a choice structure, the structure becomes a complexType. This complexType MUST be declared globally within the file containing it.	**Rationale:** This complies with current RosettaNet Practice.
Rule 4-14: The standardized structure for Property Definition Set types MUST be a complexType with the common elements declared above and with the element declarations for all the Characteristics needed for the Property Definition Set.	**Rationale**: This is standard RosettaNet practice.
Rule 5-1: Spell check SHOULD always be run on the name and definition fields. Webster MUST be used for the spelling standard. American spelling variants MUST be used instead of British (i.e., color instead of colour).	**Rationale**: As a community-agreed specification, its content should be expected to be void of spelling errors.

Rule 5-2: As a community-agreed specification, its contents SHOULD be expected to contain names and definitions that use legal grammar (within the limits stated above) that conveys clear and unambiguous information about the items.	**Rationale**: As a community-agreed specification, its content should be expected to be void of grammatical errors.
Rule 5-3: As long as a MajorRevision change does not dramatically alter the semantics of an item, the Code MUST remain the same.	**Rationale**: This is essential to support the scenario where the supplier has a different version of the Characteristic or EIPS than the customer, yet still believes the information to be exchanged will be meaningful to the customer. The commonality of the code, in spite of MajorRevision differences, will allow the supplier to do this. However, if the changes made to an item are significant enough such that confusion would result in this scenario, then an entirely new item should be created instead, with a different code.
Rule 5-4: Tag content MUST NOT be enclosed in single or double quotes, parentheses, or any other delimiters, except for a trailing period or other appropriate terminal punctuation in the case of a definition or remark.	**Rationale**: The starting and terminating tags are the expected delimiters. Any other delimiter just reduplicates the XML delimiters.
Rule 5-5: The content of a ReferenceSource, or other element, whose semantics do not demand a complete or partial sentence, MUST not be terminated with a punctuation mark (such as a period) unless that character is an integral part of the value itself.	**Rationale**: Punctuation marks that have no semantic meaning waste space and confuse the reader.
Rule 5-6: The MajorRevision number of an item MUST be incremented by one and the MinorRevision MUST be set to zero in the case where changes are made to any part of the containing item that cannot be guaranteed to be backward compatible.	**Rationale**: Users must be warned exactly which items have undergone a change that might affect the back-end mapping of the user.
Rule 5-7: For certain kinds of changes, the MajorRevision MUST always be incremented (refer to the following section 5.1.7.2). But other kinds of changes will have to be evaluated on a case by case basis to determine if the particular change would cause a compatibility issue.	**Rationale**: Since the user must evaluate whether a major revision will affect back-end mapping, major revisions should only be specified if they are truly necessary.
Rule 5-8: The version number MUST be changed if the Name changes (including minor spelling changes).	**Rationale**: Some solution providers may use the Name as the mapping key. Hence if the spelling changes the mapping must also change.
Rule 5-9: The MinorRevision of an item MUST be incremented by 1 and the MajorRevision MUST NOT be changed in the case where changes to the containing item are backward compatible.	**Rationale**: This rule will keep to a minimum the items that the user must check for back-end mapping issues.

Rule 5-10: Changes that are not simple typographical corrections to a normative element of any item MUST be evaluated to determine if the semantics are affected to the degree where backward compatibility would be compromised, and if so, a MajorRevision increment MUST be required.	**Rationale**: Any change that potentially compromises back-end mapping must be expressed as a major revision.
Rule 5-11: If the content of an EIPS/XSD has been changed, the version of the EIPS/XSD MUST also be changed. The MajorRevision of the EIPS/XSD MUST be incremented by one and the MinorRevision MUST be set to zero if either or both of the following conditions are met: ➢ If any characteristics were added to or deleted from the EIPS/XSD ➢ If the changes to the EIPS/XSD resulted in a MajorRevision for one or more of its characteristics In the case of all other changes, the MinorRevision of the EIPS/XSD MUST be incremented by 1 and the MajorRevision MUST not be changed.	**Rationale**: The versioning of the EIPS provides the user with an overview of the kinds of changes to its content. If the EIPS only shows a minor revision at the EIPS level, the user need not check further as to whether back-end mapping will be affected. However, if the EIPS shows a major revision, then the user must look at each data item in the EIPS to see whether back-end mapping has actually been affected (e.g, the user may not be using the data item that had a major revision).
Rule 5-12: The Name SHOULD match the most commonly used American English name within the industry for which the represented product is used.	**Rationale**: As a community-agreed item, the name should be the one that is most commonly used in the community.
Rule 5-13: “Upper-Case Camel” MUST be used; that is, the first letter of each word must be capitalized and the rest be lower case, unless the word is an acronym, where it would be all capitals.	**Rationale**: Current RosettaNet practice.
Rule 5-14: The Name MUST contains only alphanumeric characters. It MUST not contain any spaces or special characters.	**Rationale**: W3C practice and rules.
Rule 5-15: The content combined with the content of the MajorRevision MUST form a unique key in the namespace of the containing file.	**Rationale**: Most existing back-end Product Data Management systems use the code as the access key to the back-end system, not the Name.
Rule 5-16: The Code SHOULD, if possible, consist of three or four alphabetic characters followed by three or four digits.	**Rationale**: Three letters followed by three numbers is standard practice for all technical dictionary codes, but not all RNTD codes follow that rule.
Rule 5-17: The Definition field MUST contains at least a ShortDefinition. It MAY also contains ExtendedDefinition and Remark.	**Rationale**: A ShortDefinition is the minimum amount of information needed to recognize the semantics of a Name. However, some semantics for some properties may be quite complex and need additional information to provide complete understanding.

Rule 5-18: The definition MUST not be a simple copy of the item's name.	**Rationale**: A simple copy of the item's name adds no new information and wastes space, and an effort should be made to avoid reuse of the name at all in the definition.
Rule 5-19: All definition words MUST be spelled correctly. The combination of the short and extended definitions MUST provide a clear and unambiguous meaning for the item. Both of these definitions are normative. Remarks provide non-normative but often useful information.	**Rationale**: As previously explained in Rule 5-1 and Rule 5-2, an EIPS, as a community-agreed specification is expected to be void of grammatical and spelling errors.
Rule 5-20: The content of ShortDefinition MUST be one phrase free of any XML markup, that is, free-form text.	**Rationale**: This rule facilitates the use of the ShortDefinition in a mouse-over process.
Rule 5-21: The phrase MUST not repeat the name of the item.	**Rationale**: To repeat the name of the item in a definition wastes space and may result in a circular definition.
Rule 5-22: The semantics of the content MUST not contradict those of the ExtendedDefinition.	**Rationale**: Users will be confounded if they encounter two definitions with different meanings for the same property.
Rule 5-23: The semantics of the content MUST not contradict those of the ShortDefinition.	**Rationale**: As previously explained in Rule 5-22 immediately above, users will be confounded if they encounter two definitions with different meanings for the same property.
Rule 5-24: The additional, normative details included in the ExtendedDefinition MUST all be complete, grammatically correct sentences.	**Rationale:** As previously explained in Rule 5-2, an EIPS, as a community-agreed specification is expected to be void of grammatical errors.
Rule 5-25: Comments, notes, or other non-normative remarks about the item MAY be provided.	**Rationale**: If the 15 year history of technical dictionaries, it has proven to be useful to occasionally provide non-normative information such as how to know when a certain property should be provided with another property.
Rule 5-26: This number MUST be a two digit number with initial zeros. It is initially "01" and MUST be incremented by 1 each time this item is revised in such a way as to affect mappings using this item.	**Rationale**: The two digit number rule is current RosettaNet practice and the increment rule reflects the versioning rules in section 5.1.7.
Rule 5-27: This number is a two digit number with initial zeros. It is initially "00" and MUST be incremented by 1 each time this item is revised in a way that does not affect mappings using this item; and MUST be reset to"00" when a major revision occurs.	**Rationale**: The two digit number rule is current RosettaNet practice and the increment rule reflects the versioning rules in section 5.1.7.
Rule 5-28: This field MUST contain the date that this item was initially defined. The content MUST be a 10-character date in IS0 8601 format, e.g., CCYY-MM-DD: "1994-03-21".	**Rationale**: This is current RosettaNet practice.

Rule 5-29: This field MUST contain the date that this item was last changed. This is the latest date of either the DateOfMajorRevision or DateOfMinorRevision.	**Rationale**: To be consistent with RosettaNet practice, which does not distinguish the dates of major and minor updates.
Rule 5-30: The content MUST be a 10-character date in IS0 8601 format, e.g., CCYY-MM-DD: "1994-03-21".	**Rationale**: This is current RosettaNet practice.
Rule 5-31: This field MUST contain a 5-digit code consisting of the last two digits of the MajorRevisionNumber, followed by a period, followed by the last two digits of the MinorRevisionNumber.	**Rationale**: This is current RosettaNet practice.
Rule 5-32: This field MAY contain an abbreviated, plain text representation of the PreferredSymbol, MUST utilize Latin letters only, and MUST be constructed according to the following rules: 1. The first character MUST be a letter, "@", or a "$"when defining a specific Greek letter. If applicable, the "@" sign precedes the "$" sign. 2. The length MUST be limited to 15 characters. 3. Greek letters that are graphically identical to Latin letters are not converted. 4. Other Greek letters MUST be represented by the corresponding ISO/R843 Latin letter preceded by a dollar "$" character. 5. An asterisk "*" character MUST be used to indicate that the following letter is a superscript. 6. An underline "_" character MUST be used to indicate that the following letter is a subscript.	**Rationale**: The symbols used to identify technical properties often use subscripts, superscripts, and Greek letters. Many of these, of course, cannot be expressed in ascii (Latin) characters so they have to be converted by some standard rule.
Rule 5-33: This field MAY contain a non-normative reference to the document from which a significant portion of the semantics of the parent element was derived. (Note: No standard encoding has been defined to make the content machine-sensible.)	**Rationale**: This reference is required by our MOUs with IEC and ECALS. It also shows that we respect the intellectual property of the sources.
Rule 5-34: There MUST be a ReferenceSource for each distinct reference.	**Rationale**: If the definition was taken from IEC or ECALS, which in turn took the definition form some other ISO or IEC standard, then both must be referenced to protect the intellectual property of both parties.
Rule 5-35: This tag MAY point to a non-text object. The pointer MUST be coded in such a way the person viewing the XSD will be able to see the graphic with the most common XML viewing and editing software.	**Rationale**: Some technical properties define a complex curve that will be easier to understand if shown in a graphic rather than trying to explain curves and line angles in text alone.

Rule 5-36: The appropriate ValueFormat MUST be selected from the “Value” column of Table 5.3.	**Rationale**: As previously explained in Rule 4.8, if the correct ValueFormat is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.
Rule 5-37: The pattern MUST be expressed within Validation General Expression in Table 5.3 that corresponds to the Value defined in ValueFormat.	**Rationale**: As previously explained in Rule 4.8, if the correct general expression pattern is not selected from the corresponding part of Table 5.3, the data will not be validated correctly.
Rule 5-38: Depending on the type of data the Characteristic is defining, some of the following additional fields MAY appear.	**Rationale**: The lower left half of the data model in chapter 3 shows that measured values may consist of either a simple value or a set of three (toleranced) values or up to four min/max values.
Rule 5-39: The NameOfBasicDataFormat MUST be a name in Table 5.3 that corresponds to the contents of the ValueFormat in the same Characteristic.	**Rationale**: If the above rule is not followed the property will not validate correctly.
Rule 5-40: The ValueFormat MUST be a value in the value column of Table 5.3 that will cause the appropriate validation of the content of the Characteristic.	**Rationale**: If the above rule is not followed the property will not validate correctly.
Rule 5-41: The instance of a code in the message MUST be one of the ValueCodes provided in the value code list of this Characteristic.	**Rationale**: The value code list defines the domain of acceptable values. Anything outside of this domain is an error.
Rule 5-42: The ValueCode MUST not duplicates another ValueCode in the same list.	**Rationale**: Allowing duplicate values will confuse the computer when validating the message.
Rule 5-43: This field MUST contain a human-understandable explanation of the meaning of the ValueCode.	**Rationale**: The whole point of providing the meaning is to allow a human to understand the meaning of the value code.
Rule 5-44: The content of the Unit field SHOULD be as specified within one of the following standards for unit of measure. ➢ International System of Units (SI) (http://physics.nist.gov/cuu/Units/index.html) ➢ IEC 61360-1 (Annex B) (http://www.iec.ch/)	**Rationale**: Both are internationally agreed lists of units of measure and cover 99.8 percent of units of measure needed for electronic components.
Rule 5-45: The units of measure MAY also include the following additions to the above standards for units of measure: ppm (parts per million) and % (percent).	**Rationale**: Both of those additions are needed for technical specifications but are not part of either of the above standards.
Rule 5-46: When an appropriate SI Unit exists for the property to be defined, the Units of Measure MUST be defined using the content of the UOM tables in Section 3.2.1 (Constructing Units of Measure) of the EIPS/XSD Creation/Maintenance guideline.	**Rationale**: The tables in the *EIPS/XSD Creation/Maintenance guideline* have been extracted from the standards required by rule 5-43.

Rule 5-47: Units not covered by Section 3.2.1 (Constructing Units of Measure) of the *EIPS/XSD Creation/Maintenance guideline* MUST have a "1" coded in the Unit field, and the item's short definition, and optionally its name, MUST specify the meaning of the measurement.	**Rationale**: International standards typically cover all scientific units of measure but not all engineering units of measure. Take speed for an example: there is an SI unit for physical distance covered over time but no SI unit to explain the speed of a CPU (which does not move over some physical distance but rather is computing a number of instructions over time). These engineering units of measure are ignored by ISO and IEC but we need to define them. In that case, there is no ISO or IEC unit to be coded in the unit field, so the unit is replaced with a "1".
Rule 5-48: The unit of measure for an item MUST be consistent with the name and definition of that item. For example, if the name or definition refers to a voltage or current value, then the unit of measure MUST be in volts or amperes (respectively).	**Rationale**: A community-agreed standard should not confuse users with units of measure that contradict the name or definition of the Characteristic.
Rule 5-49: Units that consist solely of alphanumeric characters, including characters that can be represented as XML character entities, MAY be entered literally. However, if the symbol consists of any math operation (such as explicit or implied multiplication, division, or exponentiation, etc.), or any display format information (such as subscripting, superscription, overbar, prime, etc.), then the symbol must be encoded in MathML (Presentation style).	**Rationale**: Units of measure are often expressed in mathematical expressions. MathML is a well-known W3C standard for expressing mathematical expressions.
Rule 5-50: The Characteristics in the EIPS/XSD MUST be coded in alphabetical order by Name.	**Rationale**: This is current RosettaNet practice.
Rule 6-1: When a technical data object is created the major revision number is 01 and the minor revision number is 00.	**Rationale**: This is current RosettaNet practice.
Rule 6-2: If a change to EIPS/XSD content MAY force current users to make changes in their coding or tables, then the change MUST be a major revision. Otherwise the change MUST be a minor revision.	**Rationale**: As explained in the Versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.
Rule 6-3: If the change does not change any current functionality but enables new functionality and is optional, it will not affect current coding or tables unless the user wants that new functionality. This kind of change MUST be a minor revision.	**Rationale**: As explained in the Versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.
Rule 6-4: Version numbers MUST be two digits long with leading zeroes.	**Rationale**: This is current RosettaNet practice.

Rule 6-5: For minor revisions of items, the MinorRevisionNumber is increased by 1. For major revisions, the MajorRevisionNumber is increased by 1 and the MinorRevisionNumber is reset to 00.	**Rationale**: This is current RosettaNet practice.
Rule 6-6: A major revision MUST occur when any change is made to the Name.	**Rationale**: Some solutions MAY use the Name as the mandatory access point to the EIPS.
Rule 6-7: A major revision MUST occur when non-editorial changes are made to the ID, definition, or domain ("type," Units, ValueFormat, or ValueDomain).	**Rationale**: Many solutions use the ID as the access point to the EIPS and to the backend system. Non-editorial changes to the definition MAY change the scope of its semantics and thus the mapping to the backend system. Changes to the domain MAY require changes to subroutines that convert the data in the message instance before it is sent to the backend system.
Rule 6-8: Obsolescing MUST result in a major revision.	**Rationale**: If a user is using this item, then the user's coding and/or tables must be revised.
Rule 6-9: All other changes MUST be minor revisions.	**Rationale**: Users who previously used the property called Supply Voltage in the classes where the scope of meaning was divided between single supply voltage and dual supply voltage will have to recode their mapping to include either or both of these new properties with their new codes. However, users of the same property in classes where the original concept still applies can continue to use the old property and need not update their mappings.
Rule 6-10: A major revision MUST occur when any change is made to the Name.	**Rationale**: Some solutions MAY use the Name as the mandatory access point to the EIPS.
Rule 6-11: A major revision MUST occur when non-editorial changes are made to the ID or Definition. The Definition is said to change if its scope of application changes.	**Rationale**: Many solutions use the ID as the access point to the EIPS and to the backend system. Non-editorial changes to the definition MAY change the scope of its semantics and thus the mapping to the backend system.
Rule 6-12: Obsoletion MUST result in a major revision.	**Rationale**: If a user is using this item, then the user's coding and/or tables must be revised.
Rule 6-13: A major revision MUST occur when a change is made in the sequence of Characteristics currently defined in the EIPS. Adding Characteristics to the end of the EIPS is a minor revision.	**Rationale**: This rule is required because the sequence of Characteristics is defined as an "ordered" list of CharacteristicDefinitions, and solution providers may have coded a dependence on the ordering. Since the items are all optional (but ordered), additions at the end should not affect current coding.
Rule 6-14: All other changes MUST be minor revisions.	**Rationale:** As explained in the versioning rules in section 5.1.7, users must be given clear indication of which items have undergone a change and whether that change is major or minor, so that they can review their backend mapping to see whether it has been affected by the change.

Rule 6-15: If any item in a given EIPS file undergoes a major revision during a maintenance cycle, the next release of that EIPS file MUST be a major revision.	**Rationale**: As explained in the Versioning rules in section 5.1.7, users must be warned that some items have undergone a change which might affect the back-end mapping of the user.
Rule 6-16: If the file undergoes a major revision, the versioning of the file name MUST follow RosettaNet rules for the versioning of files.	**Rationale**: As explained in the Versioning rules in section 5.1.7, versioning of technical information items should follow RosettaNet rules to minimize errors.
Rule 6-17: Obsolete items MUST be kept in the released EIPS file for one release cycle and then not output in any later releases.	**Rationale**: Although an item has been obsolesced, the user will have one maintenance cycle within which to update the mapping tables to the new version of the EIPS. In the meantime the old item can still be used if both partners agree.
Rule 6-18: Items to be obsoleted MUST be described in the accompanying release documentation.	**Rationale**: Although an item has been obsolesced, the user will have one maintenance cycle within which to update the mapping tables to the new version of the EIPS. In the meantime the old item can still be used if both partners agree.
Rule 7-1: The EIPS/XSD file name MUST comply with the following requirements: Requirement 7-1-1: The EIPS/XSD applies to MUST be easily identified. Requirement 7-1-2: The EIPS/XSD file name MUST identify its versioning (major and minor version numbers.	**Rationale**: Compliance with RosettaNet practices.
Rule 7-2: The EIPS/XSD MUST comply with TPIR-PIP namespace and namespace prefix rules.	**Rationale:** Since the TPIR-EIPS is created based upon a TPIR-PIP, the TPIR-EIPS namespace and namespace prefix must comply with TPIR-PIP rules.
Rule 7-3: The subtype field in the EIPS/XSD namespace specific string MUST be “eips.”	**Rationale:** By setting the subtype to eips, the application can deduct that the file contains an EIPS/XSD.
Rule 7-4: The EIPS/XSD namespace prefix MUST start with an "e" for EIPS then followed by the initial letter of every word in the name. E.g., If the name is Capacitor-Fixed-Tantalum-Solid, the the namespace prefix would be “ecfts”. If the prefix duplicates another existing prefix, then the duplication is eliminated by adding a suffix letter, starting with “a” and continuing through the alphabet until no duplication exists.	**Rationale:** The namespace prefix should be generated using simple rules and should be persistent for a specific EIPS.
Rule 7-5: Each EIPS MUST be contained in its own separate file.	**Rationale**: EIM program requirement. It also reinforces modularity for all TPIR PIPs with Engineering Information.
Rule 7-6: The file MUST contain all element declarations in alphabetic order, and all the EIPS complexType definitions containing all the element declarations in alphabetic order.	**Rationale**: This structure and sequencing is standard RosettaNet practice.

Rule 7-7: Each EIPS MUST define its own namespace. The namespace prefix must be persistent.	**Rationale**: RosettaNet utilizes a homogenous Namespace concept. The persistence of a namespace prefix simplifies implementation of multiple EIPS by implementers. It also works as a unique identifier for XML Schema files inside RosettaNet.
Rule 7-8: Each of the EIPS delivery packages MUST have the following structure and content.	**Rationale**: This structure complies with the structure and contents of the latest RosettaNet PIP delivery packages, ignoring items, such as the BPSS, that do no apply to an EIPS.
Rule 7-9: The EIPS/XSD file MUST be stored under the Design folder.	**Rationale**: These definitions are only used in the domain of design engineering and are not used in any other domain.
Rule 7-10: The EIPSBasicFormat.xsd MUST be included with the EIPS.xsd file.	**Rationale**: The use of the basic format file increases the modularity of TPIR PIPs with Engineering Information and ensures that each EIPS has the correct version of the basic format file.