

Energistics Packaging Conventions Specification v1.1

For use with Energistics data-transfer standards

EPC Overview	To address the challenges of the multi-file data sets used in upstream oil and gas, Energistics and its members have developed file packaging conventions based on the Open Packaging Conventions (OPC), a widely used container-file technology that allows multiple types of files to be bundled together into a single package—which is essentially a zip file, with a couple of extra requirements. The Energistics Packaging Convention (EPC) may be used with all Energistics domain standards.
Version of Standard	1.1
Version of Document	1.0
Date published	May 16, 2022
Prepared by	Energistics
Abstract	This document defines and describes the details of the Energistics Packaging Conventions (EPC).
Document type	Specification
Language	U.S. English
Keywords:	standards, energy, data, information, process, file packaging convention



Usage, Intellectual Property Rights, and Copyright

This document was developed using the Energistics Standards Procedures. These procedures help implement Energistics' requirements for consensus building and openness. Questions concerning the meaning of the contents of this document or comments about the standards procedures may be sent to Energistics at info@energistics.org.

The material described in this document was developed by and is the intellectual property of Energistics. Energistics develops material for open, public use so that the material is accessible and can be of maximum value to everyone.

Use of the material in this document is governed by the Energistics Intellectual Property Policy document and the Product Licensing Agreement, both of which can be found on the Energistics website, <https://www.energistics.org/legal-page/>.

All Energistics published materials are freely available for public comment and use. Anyone may copy and share the materials but must always acknowledge Energistics as the source. No one may restrict use or dissemination of Energistics materials in any way.

Trademarks

Energistics®, Adopt>Advance>Accelerate®, Energistics Certified Product® and their logos are registered trademarks and WITSML™, PRODML™, RESQML™ are trademarks or registered trademarks of Energistics Consortium, Inc. in the United States. Access, receipt, and/or use of these documents and all Energistics materials are generally available to the public and are specifically governed by the Energistics Product Licensing Agreement (<http://www.energistics.org/product-license-agreement>).

Other company, product, or service names may be trademarks or service marks of others.

Amendment History			
Standard Version	Document Version	Date	Comment
1.1	1.0	May 16, 2022	<p>Revisions have been made due in part to changes in the format of Energistics Identifiers (for requirements for Energistics Transfer Protocol (ETP)). While EPC is available for use by all Energistics domain standards, RESQML uses EPC most often, so the Jira tickets were created by the RESQML SIG.</p> <ul style="list-style-type: none">• RESQML-569 and RESQML-589. Removed EpcExternalPartReference (a top-level element) and replaced the functionality with ExternalDataArray, which is a complex type that is part of the data object that is referencing data in an external file (i.e., not in the data object file). EpcExternalPartReference was removed from Section 3.2. Content to explain use of the ExternalDataArray was added in Section 3.2.3.1. This change also impacted relationship names in Section 3.4.• RESQML-515. Specifies the rels entry for Energistics Property Kind Dictionary. See Section 3.4.1.• RESQML-598. Specifies the preferred mechanism for creating DataObjectReferences (DOR) in CustomData (data objects that are not defined by an Energistics domain standard). See Section 3.4.1.

Table of Contents

Table of Contents.....	4
1 Introduction.....	5
1.1 What is Energistics Packaging Conventions (EPC)?	5
1.1.1 Challenge: Complex, Multi-File Datasets	5
1.1.2 Solution: Standards for Packaging Together Files.....	5
1.1.3 Why OPC?.....	5
1.2 Audience, Purpose, and Scope	6
1.2.1 EPC Scope	6
1.3 Documentation Conventions.....	6
1.4 Resource Set	6
1.4.1 OPC Resources.....	7
2 OPC Overview	8
2.1 Basic Characteristics	8
2.2 OPC Key Concepts.....	8
2.2.1 Content Type	9
2.2.2 Relationships	9
3 EPC-Specific Content and Structure	11
3.1 Package Naming Convention	11
3.2 The EPC Parts	11
3.2.1 Part Naming Conventions and Content Types	12
3.2.2 XML Data Objects	12
3.2.3 Storing Files Externally from the EPC File	13
3.3 Content Types.....	13
3.4 Relationships	14
3.4.1 Rules for Specifying Relationships.....	15
3.4.2 Relationship Types and Namespaces.....	15
3.5 Core Properties	16
3.5.1 Existence	16
3.5.2 Location	16
3.5.3 Contents	16
3.6 Directory Layout.....	17

1 Introduction

1.1 What is Energistics Packaging Conventions (EPC)?

The Energistics Packaging Conventions (EPC) are based on the Open Packaging Conventions (OPC), a widely used container-file technology that allows multiple types of files to be bundled together into a single package. Essentially, it's a Zip file specifically tailored for use with Energistics domain standards.

EPC is part of the Energistics Common Technical Architecture (CTA), a set of shared components and technologies designed for and adopted for use with Energistics domain standards.

1.1.1 Challenge: Complex, Multi-File Datasets

With the complexity of oil and gas exploration and production (E&P), petro-technical professionals find themselves dealing with large, complex data sets, which may include various types of data files, reports, images, and more.

When exchanging that data during the course of normal E&P workflows and through the use of related software applications, users need to know that they do indeed have "all the pieces" that make up a particular data set, and they need to understand what those pieces are and how they are related to one another.

1.1.2 Solution: Standards for Packaging Together Files

To address these challenges, Energistics and its members have developed a file packaging convention based on the Open Packaging Conventions (OPC), a widely used container-file technology that allows multiple types of files to be bundled together into a single package. Built on the widely used ZIP file structure and originally created by Microsoft, OPC is now an open standard supported by these standards organizations:

- Ecma International (<http://www.ecma-international.org/publications/standards/Ecma-376.htm>)
- ISO/IEC 29500-2:2012, which has 4 parts, which are all freely available at this link (near bottom of the page) (<http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>).

Energistics has adopted all of these practices and adapted them to meet the needs of its data-transfer standards, beginning with RESQML V2.0.

- This tailoring of OPC is referred to as the Energistics Packaging Conventions (EPC).
- The resulting package that contains all of the component files is referred to as an EPC file (or sometimes an Energistics package).

IMPORTANT! IT professionals implementing EPC MUST use either OPC libraries or the OPC specification for the majority of their coding. Use this EPC specification to implement the EPC adaptations, which are explained in Chapter 3 (page 11).

1.1.3 Why OPC?

Energistics chose to base EPC on OPC for several key reasons, which include:

- OPC is an existing, widely supported standard.
- It addresses the use cases and requirements of Energistics, as explained above. Namely, it specifically supports the loose coupling of multiple document parts into a coherent whole. It has the ability to appear as folder and file structure as well as a single file archive.
- It supports a rich and extensible mechanism for describing relationships between the package and individual parts or files within the package and resources external to the package.
- Various programming languages already provide wide support (read-write libraries) for OPC. These languages include: the Microsoft.NET framework and Java. Energistics SIG members have tested various tool sets, and several vendors, whose applications rely on very large 3D models and digital media, have field-tested it. These libraries handle the parts and relationships of the package.

- It is based on the commonly used ZIP file structure. In the absence of explicit support for EPC, developers can at least open the package and see the list of files using commonly available zip tools. (To open with a ZIP tool: Right-click the file; from the menu, choose "Open with"; and select or navigate to your ZIP tool.)

1.2 Audience, Purpose, and Scope

This document is intended for use by information technology (IT) professionals (e.g., software developers, architects, etc.) to help them understand, create, and read an EPC file.

The scope of this document is to:

- Provide a high-level overview of OPC (the standard on which EPC is based).
- Specify how EPC differs from OPC.

1.2.1 EPC Scope

EPC can be used with version 2 and above of Energistics domain standards (RESQML, WITSML, PRODML, etc.).

As of this publication, the scope of an EPC file is that it is always transient. Energistics and its members are discussing use of EPC to support the notion of projects and archives. However, for this version, the focus is only on using EPC to group together files for data transfer.

1.3 Documentation Conventions

This documentation observes the conventions listed in below.

	Document/Resource	Description
1.	Document Hyperlinks: Internal	Though no special text-formatting convention is used: All section, page, and figure numbers in this and all Energistics documents are hyperlinks. The table of contents is also hyperlinked.

1.4 Resource Set

Energistics provides the following resources to help understand and implement EPC.

Document/Resource	Description
<i>Energistics Packaging Conventions Specification</i> (this document)	Rules and guidelines for implementing EPC and a brief overview of OPC, on which EPC is based.

For information about metadata standards used by Energistics, see the following:

Document/Resource	Description
<i>Energy Industry Profile of ISO 19115-1 (EIP)</i> Link to Web page: http://www.energistics.org/asset-data-management/energy-industry-profile-standard	An open, non-proprietary transfer standard for metadata used to document information resources, and in particular resources referenced to a geographic location, e.g., geospatial datasets and web services, physical resources with associated location, or mapping, interpretation, and modeling datasets. The EIP is an ISO Conformance Level 1 profile of the widely adopted international standards ISO 19115-1:2014 which provides XML implementation guidance with reference to ISO Technical Specification 19115-3:2016.

1.4.1 OPC Resources

The table below contains links to the OPC specifications published by Ecma International and ISO.

	Document/Resource	Description
1.	<i>Standard ECMA-376 Office Open XML File Formats</i>	http://www.ecma-international.org/publications/standards/Ecma-376.htm
2.	<i>ISO/IEC 29500-2:2012 Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 2: Open Packaging Conventions</i>	http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html (All four parts of the standard are freely available at this link; scroll to bottom of the page.)

2 OPC Overview

Because EPC is based on OPC, this chapter has been provided to explain some general concepts about how OPC works.

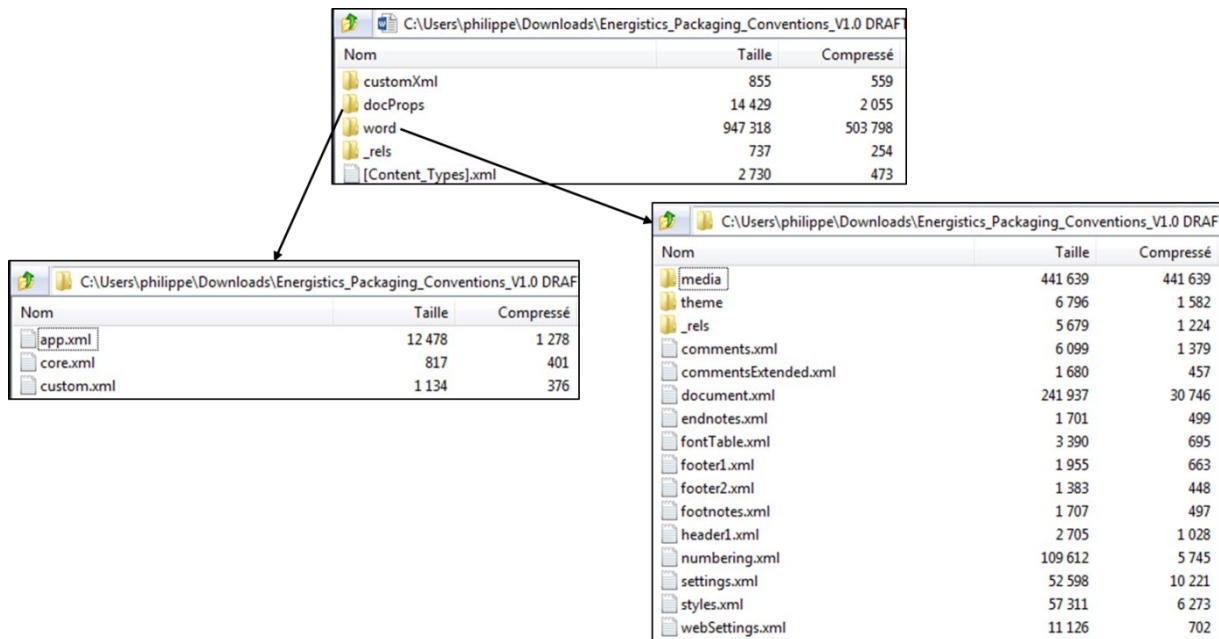
2.1 Basic Characteristics

The basic characteristics of an OPC package are:

- It is a ZIP archive containing various files called *parts* with different (informed) content.
- It may be documented by means of metadata called *core properties*.
- Relationships between parts are formally described.

2.2 OPC Key Concepts

This section provides an overview of key OPC concepts that are important to understanding how an OPC package is structured and how it works. **Figure 1** shows an example of an OPC package. The table below lists and describes the key concepts and includes references to more information available in this document.



Nom	Taille	Compressé
custom.xml	855	559
docProps	14 429	2 055
word	947 318	503 798
_rels	737	254
[Content_Types].xml	2 730	473

Nom	Taille	Compressé
app.xml	12 478	1 278
core.xml	817	401
custom.xml	1 134	376

Nom	Taille	Compressé
media	441 639	441 639
theme	6 796	1 582
_rels	5 679	1 224
comments.xml	6 099	1 379
commentsExtended.xml	1 680	457
document.xml	241 937	30 746
endnotes.xml	1 701	499
fontTable.xml	3 390	695
footer1.xml	1 955	663
footer2.xml	1 383	448
footnotes.xml	1 707	497
header1.xml	2 705	1 028
numbering.xml	109 612	5 745
settings.xml	52 598	10 221
styles.xml	57 311	6 273
webSettings.xml	11 126	702

Figure 1: An example OPC file. Microsoft uses OPC to group together the parts that make up documents (.docx files). This example is the OPC file for the specification document you are currently reading.

OPC Concept	Definition, Purpose, and Requirements
[Content_Types].xml	<p>This file contains the content type, a description of the type of content of a part, for each part in the package.</p> <p>This file name is a reserved name in OPC.</p> <p>For:</p> <ul style="list-style-type: none"> General OPC information, see Section 2.2.1 (below). EPC-specific information, see Section 3.3 (page 13).
Relationship folders and files (_rels)	<p>Parts may contain references to other parts in the package and to resources outside of the package.</p> <p>Uses _rels folders and .rels extensions, which are reserved for this purpose.</p> <p>For:</p> <ul style="list-style-type: none"> General OPC information, see Section 2.2.2 (below). EPC-specific information, see Section 3.4 (page 14).
Core Properties	<p>The identifying information of a package, which includes key metadata such as the creator of the package and the date it was created.</p> <p>Core properties are now optional in OPC, but are required for EPC.</p> <p>For EPC-specific information, see Section 3.5 (page 16).</p>

2.2.1 Content Type

The root folder of an OPC package contains a special file with the reserved name *[Content_Types].xml*; this file **MUST** always be stored in the root. The purpose of this file is to define a specific mime content type for each part in the package. For more information on how this is used in EPC, see Section 3.3 (page 13).

The mime-type mapping allows a level of indirection from file extensions (which can have different meanings in different systems) and an explicit type that software packages **MUST** support. It can also be useful:

- For XML content: if you have multiple versions of RESQML inside the same EPC container, you can use the mime type to determine which documents to validate.
- For non-XML content (such as PDF or application files, such as .docx), it provides a hint to the operating system about which viewers and editors support the content.

2.2.2 Relationships

OPC provides mechanisms for defining relationships between the package parts, and external resources. The mechanism can be used for both internal parts and external resources, meaning that relationships can actually point outside the package, which includes a file on a Web or file server, a Web service address, or almost anything that can be described by a URI.

Each folder of the package (including the root), has a special folder with the reserved name “_rels”. This folder contains files whose names are made up of any file name in the parent folder, with the reserved extension “.rels” appended to it.

The relationships for the package as a whole are defined in the file */_rels/.rels*. (This is commonly where the location of the core properties file is stored.) The following table explains the key components of the .rels file.

Attribute	Definition/Purpose
Id	A unique name for the relationship. It MUST be unique within the context of the overall package.
Target	The file or resource to which the relationship refers. It can point to a part inside the package or to an external resource.
Type	Indicates the kind of relationship that exists.

3 EPC-Specific Content and Structure

An EPC file (or Energistics package) is an OPC file; this means first and foremost that any standard OPC reader or library should be able to read an EPC file. This chapter explains the EPC-specific differences that developers **MUST** address when implementing EPC.

IMPORTANT! IT professionals implementing EPC must use either OPC libraries or the OPC specification for the majority of their coding. Use this EPC specification to implement the EPC adaptations, which are explained in this chapter.

Figure 2 shows an example EPC file.

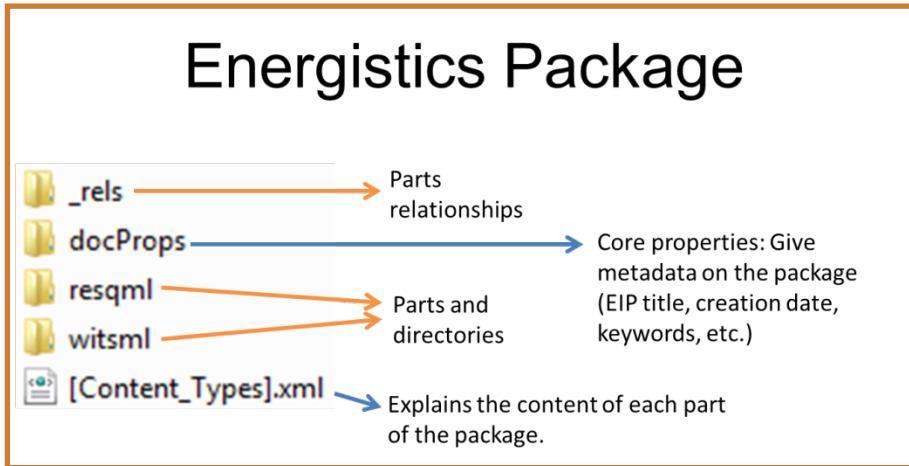


Figure 2: An example EPC file. The basic package structure is consistent with OPC; that is; the root folder **MUST** contain at least the **_rels** folder and the **[Content_Types].xml** document. An EPC file **MUST** include core properties, which may be stored in a **docProps** folder. This example includes the data being transferred in folders labeled **resqml** and **witsml**. All of these **parts** are described below.

3.1 Package Naming Convention

The EPC file **MUST** have the file extension: **.epc**

EXAMPLE: *MyProjectFilesForTransfer.epc*

3.2 The EPC Parts

The table below lists the additional parts (in addition to the standard OPC parts) used in an EPC file.

EPC Part	Definition, Purpose, and Requirements
XML data objects	<p>XML data objects defined in any of the Energistics data-transfer standards, such as RESQML, WITSML or PRODML, may be included. In Figure 2 above, these parts are stored in the <i>resqml</i> and <i>witsml</i> folders.</p> <p>NOTE: An EPC file may have xml files from multiple versions of the Energistics standards (RESQML, WITSML, and PRODML). EXAMPLE: An EPC file can contain xml files saved in RESQML v2.0.1 and v2.1.</p>
Other types of files	<p>Optionally, other files that contain additional, informal information relevant to the contents of the package, such as these listed below. As a guideline these files are stored in a folder named "media"; for more information, see Section 3.6 (page 17).</p> <ul style="list-style-type: none"> • PDF • Video • SEG Y

EPC Part	Definition, Purpose, and Requirements
	<ul style="list-style-type: none"> Images Microsoft Word documents

3.2.1 Part Naming Conventions and Content Types

Parts in an EPC package MUST follow the OPC specification on part names. EPC XML data objects have additional requirements on their names:

- The name of an EPC XML data object part MUST be composed of the object type and the UUID of the object as in: <objectType>_<uuid>.xml. **EXAMPLES:**
 - IJKGridRepresentation_a8f023cf-de55-47ac-bfb1-b81626f47f6c.xml
 - LocalEngineering2dCrS_afd24701-17fc-4e15-8d88-adc537ead8f7.xml.

The object type of an object is defined to be the XSD complexType of the root element of the data object.

NOTE: In RESQML2.0, the object type is always prefixed with "obj_".

Following OPC requirements, each part in the EPC MUST have a mime content type. The EPC requirement is that, in addition to the object type, the content type of each XML data object contains the name and version of the standard. For a description of the format for the content type, see Section 3.3 (page 13).

3.2.2 XML Data Objects

XML data objects define the actual data that is being transferred. These data objects may be from any of the Energistics data-transfer standards. For special rules for using WITSML 1.4.1 data objects with EPC, see Section 3.2.2.1 (below).

NOTE: An EPC file may have XML files from multiple versions of the Energistics standards (RESQML, WITSML, and PRODML). **EXAMPLE:** An EPC file can contain xml files saved in RESQML v2.0.1 and v2.2 and WITSML v1.4.1.1 and v2.0.

XML data objects have the following mandatory elements:

- UUID.** This is a unique ID scoped to the creator of the package. This means that the ID should be unique for all packages generated by the same creator. For more information about the format and requirements for UUIDs in Energistics standards, see the *Energistics Identifier Specification v5.0*.
- Metadata citation group,** which is an ISO 19115 EIP-derived set of metadata attached to all specializations of AbstractObject to ensure the traceability of each data object. The citation group includes data such as a title (human readable identification of the data object), and creation and update information. For more information about metadata citations in Energistics standards, see *Energistics Common Technical Architecture Overview Guide*.

NOTE: A data object may optionally just be referenced through a data object reference (DOR) and not be present in the EPC file. For example, RESQML (v2.0 or higher) allows reference to data objects to establish or maintain accurate context for a model, but the actual data may have been transferred previously or may be transferred in the future or may be accessible by a different means than an EPC file.

3.2.2.1 Rules for Using WITSML v1.4.1 Data Objects in an EPC file

When this EPC specification was first developed, the most-used version of WITSML was v1.4.1.1, which has different requirements for identifying and storing data objects. (For rules on WITSML data-object identification, see Section 2.2 of the *WITSML STORE Application Programming Interface* specification.) Therefore, the following additional rules MUST be observed when using WITSML v1.4.1.1 data objects in an EPC file:

- Each file created in WITSML MUST contain only one business object, e.g., a single well, wellbore, log, etc. This rule is so that these individual business objects can each be correctly referenced.

- WITSML UUIDs are mandatory for each data object. In an EPC file, the WITSML UUID serves the role of a UUID as specified in this EPC document. Currently use of UUIDs in WITSML is recommended but not required. For EPC, use of UUIDs is strongly recommended.

3.2.3 Storing Files Externally from the EPC File

In some cases, it may be desirable to store files externally from an EPC file. That is, the content of these files are related to the data objects stored in the EPC file, but the files are not stored in the EPC file. Examples of these types of files include the following:

- **HDF5 files**, store large explicit arrays and numerical data in Energistics domain standards.
- **Standard Energistics dictionary files**. For more information about dictionary files, see Section 3.4.1.

3.2.3.1 HDF5 Files

Hierarchical Data Format (HDF) is a data model, library, and file format for storing and managing data. It supports an unlimited variety of data types, and is designed for flexible and efficient I/O and for both high volume and complex data—particularly when compared to XML. HDF version 5 is part of the Energistics Common Technical Architecture and is used in RESQMLv2+ and with other Energistics standards.

The Energistics package is designed for data streaming, and in some implementations, has limitations in the amount of data which may be included. In contrast, HDF5 file is designed for large data files, random access (not streaming), and can already compress its data sets. As a consequence, some Energistics standards (e.g., RESQML) require that HDF5 files be stored outside the Energistics package. To accurately maintain all relationships, the package requires use of an external reference to the HDF5 file.

The following items describe how to store and reference HDF files in the context of an EPC file:

- HDF5 files may be stored inside or outside the EPC file.
- When HDF5 files are used with Energistics standards they **MUST** use the file extension: *.h5*
- If stored inside the EPC file, the mime type for an HDF5 file is: *application/x-hdf5*
- If stored outside, it is recommended that the HDF5 files are stored in the same location as the EPC file.
- If stored outside, each data object that references an HDF5 file **MUST** have an `ExternalDataArray`, which is a concatenation of `ExternalDataArrayParts`, which are pointers to a whole or to a sub-selection of an HDF5 dataset in an HDF5 file.
 - Each data object that references an HDF5 file **MUST** have a relationship file that contains an entry for each of its physical HDF5 files with the attribute “Target” set to its relative file name.
 - The corresponding entries in the relationship file **MUST** have a “type” attribute set to: <http://schemas.energistics.org/package/2012/relationships/externalResource> and the target mode set to “External”.
 - **RECOMMENDATION:** If a single HDF5 file is pointed to, then set the “Id” attribute to “Hdf5File”.
- You may have an array that is so large that it will not fit on a physical disk; therefore, a single array may reference multiple HDF5 files. A single `ExternalDataset` could therefore contain multiple `ExternalDatasetParts`, each of which references a single HDF5 dataset in a (different) HDF5 file.

3.3 Content Types

The file named `[Content_Types].xml` is used to associate file name extensions used in the package with specific mime content type. For general information about Content Types in OPC, see Section 2.2.1 (page 9). Some examples of how mime type can be used:

- For XML content: If you have multiple versions of an Energistics ML file inside the same EPC container, you can use the mime type to determine which documents to validate (i.e., if you don't support a certain version of an Energistics standard, then you don't have to validate those files).

- For non-XML content (such as PDF or application files, such as .docx): the mime type provides a hint to the operating system about which viewers and editors support the content.

The following code sample from RESQML shows examples of content types specified for an EPC file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default
    Extension="rels"
    ContentType="application/vnd.openxmlformats-package.relationships+xml" />
  <Override
    PartName="/docProps/core.xml"
    ContentType="application/vnd.openxmlformats-package.core-properties+xml"/>
  <Override
    PartName="/namespace_resqml22/IjkGridRepresentation_2aec1720-fa3e-11e5-a116-
0002a5d5c51b.xml"
    ContentType="application/x-resqml+xml;version=2.2;type=IjkGridRepresentation"/>
</Types>
```

Use these rules to define content in the EPC file:

- [ContentTypes].xml is mandatory in OPC, so is mandatory in EPC.
- An optional media folder may contain any type of media file—no restrictions. For example, it can be used to store any type of supporting documents you might want to include in an EPC file, such as graphics, videos, reports, etc.).
- Each XML data object part MUST have a describing 'contentType'. The content type MUST follow rfc 2616: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.7> and MUST have the following format:

application/x-**<energisticsStandard>**+xml;version=**<versionNumber>**;type=**<objectType>**

where:

- energisticsStandard is the name of the standard used, e.g. “resqml” or “witsml” or “prodml”
- versionNumber is the major schema version number used, e.g. “2.0”
- objectType is the XML Schema type of the root element of the XML data object

EXAMPLES:

```
"application/x-resqml+xml;version=2.2;type=FaultInterpretation"
"application/x-resqml+xml;version=2.2;type=TriangulatedSetRepresentation"
```

3.4 Relationships

EPC depends strongly on the OPC mechanism for defining relationships between parts. It is primarily used to store the relationships between various XML data objects and between data objects and external files. For general information about specifying relationships in OPC, see Section 2.2.2 (page 9).

This section provides rules and guidelines for specifying relationships in an EPC file. The following code is an example _rels file, showing the relationship between a Boundary feature and its interpretation, an activity where it is involved and its graphical information.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="ac12dc12-4951-459b-b585-90f48aa88a5a"
    Type=http://schemas.energistics.org/package/2012/relationships/sourceObject
    Target="../namespace_resqml22/HorizonInterpretation_ac12dc12-4951-459b-b585-
90f48aa88a5a.xml"/>
  <Relationship Id="a3fd3681-45c5-404f-ad8f-c7698148c82f"
    Type=http://schemas.energistics.org/package/2012/relationships/sourceObject
    Target="../namespace_eml23/Activity_a3fd3681-45c5-404f-ad8f-c7698148c82f.xml"/>
  <Relationship Id="be17c053-9189-4bc0-9db1-75aa51a026cd"
    Type=http://schemas.energistics.org/package/2012/relationships/sourceObject
    Target="../namespace_eml23/GraphicalInformationSet_be17c053-9189-4bc0-9db1-
75aa51a026cd.xml"/>
```

</Relationships>

3.4.1 Rules for Specifying Relationships

Observe the following rules and guidelines when specifying relationships in EPC:

- Every XML data object that contains an element of type `DataObjectReference` is required to have that relationship documented within the OPC relationship system.
- The relationship file **MUST** define all the forward and backward relationships of the XML data object. So for every relationship between two XML data objects there are two .rels files; one for the source object and one for the target object.
- Relationships to non-XML data objects can be documented but are optional.
- If an XML element contained in an EPC file is involved in one conceptual relationship, then the EPC file part **MUST** have an associated rels file. These XML files can be from any of the Energistics data-transfer standards including: RESQML, WITSML, and PRODML. Rels files for other EPC file parts (jpg, pdf, etc.) are optional.
- The ID of the relationships follows the OPC format: it is an XML id. It **MUST** start with a letter or an underscore (not a digit). It is acceptable for the ID to always start with an underscore.
- The Type of a relationship **MUST** be one of the Energistics defined types as given in Section 3.4.2 (below).
- If you want to reference a data object through data that is not part of an Energistics data model (i.e., data objects that are defined by CustomData), you should use a DOR block inside CustomData. You **SHOULD NOT** use extra metadata to specify the relationships. For more information about CustomData, see the *Energistics common Technical Architecture Overview Guide v2.3*.
- **Energistics Dictionary File.** Energistics defines a standard dictionary, the Energistics Standard Property Kind Dictionary, which may be used by all of the Energistics domain standards. To specify the external rel entry, use the URI for the version of the dictionary that is being referenced;
EXAMPLE:

- For the Energistics Property Kind Dictionary included with Energistics *common* v2.3, use:
<https://www.energistics.org/download-standards/>

NOTE: Implementations should not resolve these URIs but use them as an identifier, which allows an implementation to resolve these rel entries locally.

3.4.2 Relationship Types and Namespaces

The relationship type defines the role of the relationship. Energistics defines the following valid types.

The fully qualified relationship type is defined as:

`http://schemas.energistics.org/package/2012/relationships/<Type>`

where <Type> is one of the listed types in the following table.

Type	Role
destinationObject	The object in Target is the destination of the relationship.
sourceObject	The current object is the source in the relationship with the target object.
mItoExternalDataArray	The target object is a proxy object for an external file.
externalDataArrayToMl	The current object is used as a proxy object by the target object.
externalResource	The target is a resource outside of the EPC package. Note that TargetMode should be "External" for this relationship.

Type	Role
destinationMedia	The object in Target is a media representation for the current object. As a guideline, media files should be stored in a "media" folder in the root of the package.
sourceMedia	The current object is a media representation for the object in Target.
chunkedPart	The target is part of a larger data object that has been chunked into several smaller files.

3.5 Core Properties

3.5.1 Existence

Contrary to the OPC standard, the Core Properties part is mandatory in an EPC file. The Core Properties part contains key identification metadata.

3.5.2 Location

The location of the Core Properties part is specified in the OPC standards in chapter 11 as being the target of a well-defined package relationship. This means that the `/_rels/.rels` file in an EPC package MUST contain at least the following relationship:

```
<Relationship
  Id="<unique id>"
  Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties"
  Target="<location>"/>
```

Where:

- `<unique id>` is the identifier of the relationship (only needs to be unique within the relationship file).
- `<location>` is the location of the core properties part within the EPC file. A suggested location is in the "docProps" folder.
- As specified in the OPC specifications, the content type of a Core Properties part is: `application/vnd.openxmlformats-package.core-properties+xml`

3.5.3 Contents

For an EPC file, the Core Properties part should contain:

- creator (mandatory). Free text.
- created (mandatory). This field MUST correspond to the package creation time, given in the W3CDTF format.
- version (mandatory). The EPC specification version that the package is based on (currently 1.0).
- description (optional). Free text.
- identifier (optional). URN with UUID.
 - See <http://www.ietf.org/rfc/rfc2141.txt>
 - Example : `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`
- keywords (optional). Free keyword.
 - If various keywords are given in one unique string (possibly caused by the mixed attribute being set to "true" in the complex type "CT_Keywords"), then it is assumed that the delimiter between keywords is a semicolon. In this situation, trailing and leading spaces are not allowed between the key words.
- title (optional). Free text.
- All other core properties specified in OPC are assumed NOT to be given.

3.6 Directory Layout

This EPC specification imposes no additional rules for directory layout. However, as a guidelines:

- A data object XML file **MUST** be contained in a unique path.
- If there is ambiguity about a data object's namespace (i.e., the version of the standard) or if the writer wants, the unique path can contain a directory called "namespace_id" (where id is the identifier for the namespace). EXAMPLES:
 - For RESQML v2.2 files, a "namespace_resqml22" directory
 - For WITSML2.0 files, a "namespace_witsml20" directory
- If this data object is versioned, the unique path should contain a directory called "version_id" (where id is the identifier for the data object version).
- The order of these directories does not matter.
- If the writer of the EPC file wants, there can be other directories; however, they **MUST NOT** start with any of the prefixes stated above.
- If necessary, use URI encoding (%) to escape special characters.