Palantir™

Palantir XML Reference

Version 2.0.2
August 2008

Legal Statements

These legal statements are provided to allow the recipient ("You" or "Your") to use the software, the documentation, and the other materials and products ("Materials") provided by Palantir Technologies, Inc. ("Palantir") while protecting the Materials from unauthorized use or disclosure.

You agree that You shall take all reasonable measures to protect the secrecy of and avoid the unauthorized use or disclosure of the Materials provided by Palantir that You use or that are in Your possession in order to prevent them from falling into the public domain or into the possession of persons other than those persons authorized under a specific written Agreement with Palantir. Such measures shall include, but shall not be limited to, the highest degree of care that You utilize to protect Your own information of a similar nature, which shall be no less than reasonable care. You further agree to notify Palantir in writing of any actual or suspected misuse, misappropriation, or unauthorized disclosure of the Materials that may come to Your attention.

Palantir reserves the right to make changes to the Materials without notice. Before installing and using the software, review the readme files, the release notes, the latest version of the Palantir Installation Guide, and the End User License Agreement (EULA), all of which are available from Palantir.

Document Name: Palantir XML Reference
Document Version: Version 2.0.2
Document Part No: XMLRef-2.0.2
Release Date: August 2008

Palantir Customer Support

Contact Palantir Customer Support with questions about our product or training opportunities:

Phone: 877-247-2513
E-mail: support@palantirtech.com
Website: http://www.palantirtech.com/

If you have questions, comments, or suggestions about this or any other Palantir document, contact us at support@palantirtech.com. Your feedback is always welcomed.

Contents

1 About This Guide

This chapter describes this document, its contents, how to use it, and who it is intended for. The following topics appear in this section:

- Purpose
- Audience
- Typographical Conventions

Purpose

This reference describes the Palantir XML formats. After reading this document, you should have a good understanding of the supported formats and how to use them. You can also refer to the material in this guide as you build data import and export utilities and services for use with Palantir.

Audience

This reference is intended for users who are charged with importing large amounts of data into or out of a Palantir system. You should read this reference if you are building, working with, or developing third-party plugins or web services that import data into or out of Palantir.

This guide assumes that you:

- Attended the Palantir Ontology training class.

- Use a staging system to update and test changes to your ontology.
- Are familiar with the XML technology.

Typographical Conventions

Palantir documentation uses the following typographical conventions.

Table 1 Typographical Conventions in Documentation

Convention	Description
Unmodified text	Calibri font is used for instructions, conceptual (background) information, and reference information.
`Monospace`	`Courier New font` is used for: • System-generated messages • File names and paths to files • API syntax and examples
Bold	**Bold font** is used for: • User entries exactly as printed in the documentation • GUI controls, such as key names • Definitions of Palantir-specific and unfamiliar terms used in instructions, concepts, and reference information
Italics	*Italic font* for: • Variable entries that differ, such as passwords and hostnames • Variables in API syntax • References to other titles in the Palantir documentation set

Note: A Note is used to highlight important information that might be overlooked if it appeared as unmodified text.

Caution: A Caution is used as a warning to help you avoid problems. Because the Palantir platform is a software solution, we do not follow the ANSI and ISO guidelines for using **Caution** as a personal-injury signal word. Although software can produce unexpected or undesirable side effects, it cannot maim or kill you.

2 Introduction to the Palantir XML Formats

This chapter introduces you to the Palantir XML formats, pXML and DocXML. You should read this chapter if you are unfamiliar with the available formats or are interested in learning about their underlying concepts and practices. Finally, this chapter introduces the Palantir tools available for working with these formats.

Overview

You use Palantir XML (pXML) to model a Palantir object graph. pXML was developed to support the export and import of Palantir objects. pXML allows you to model multiple objects, their properties, and relationships in a single file.

A great majority of the data for import into Palantir consists of documents such as message traffic. DocXML was designed specifically to support import of documents into Palantir. DocXML allows you to model a single document object and its related entities.

Using pXML and DocXML you have flexibility and control when it comes to your data. This section introduces you to concepts you need when working with pXML or DocXML.

Role of the Object Model

You use pXML to export Palantir objects and their component parts to an external system or to import data from an external system into Palantir as objects. For this reason, you must be familiar with the Palantir object model to work effectively with pXML or DocXML.

The Palantir object model is an abstraction that sits between the end user and a system's physical storage. Meaning the object model provides a way to represent some set of real-world data in a familiar way without requiring understanding of or even interaction with the computer systems that maintain that data.

The object model is designed to model entities that exist as data in a system. Entities are things such as people, places, phones, and computers. Entities are also events such as meetings or arrests or parties. They are also documents such as email messages or passports. Within Palantir, entities are represented by objects.

Objects have components that describe some aspect of the object. Components can be:

- properties such as eye color, age, or name
- relationships such as brother of or went to school at
- media such as images, documents, or videos
- notes which are free text added to an object

You can use the objects and components in the Palantir object model to represent any problem domain.

The Ontology

Each Palantir installation has its own unique ontology. An ontology is simply a way of naming and arranging objects and their components such that the resulting structures makes sense to your organization and what it does. For example, a police unit would have objects such as suspect, defense attorney, arrest, police report and so forth. An military unit might have objects such as combatant, battle, topographical map, enemy camp, and so forth.

Every Palantir installation has an ontology; Palantir installs a default ontology that users can customize. You need to have a good understanding of your ontology to use pXML or DocXML effectively. For example, you have to know which document subtypes exist in your ontology. Then, you can define the appropriate document types within the XML so that, on import, the documents appear in Palantir as you would expect.

Before working with Palantir's XML formats, you should use the Palantir Ontology Manager to get familiar with your organization's ontology. Refer to the *Palantir Dynamic Ontology Manager Guide* for information on using the manager.

When to Use the XML Formats

Palantir anticipates that you will have a lot of external data you need to import from various data sources. In general, use DocXML for importing documents that are the source for entities, objects, and the relationships between them. Use pXML for data that you cannot use with the Data Importer utility, for example, if you have a database structure that includes joins across tables.

You can import structured data from a database (with some limitations) or unstructured data such as files in a file system. In general, you should use pXML and DocXML imports for situations where your organization has static legacy data in a read-only source or large archives of message-based documents.

Typically, the system expects that you will import data into Palantir and then discard the XML document you used. If your installation is integrated with one or more entity extractors, the system will extract entities from your document in addition to importing it into Palantir.

If you re-import a DocXML file, the system does not recognize the document or the entities within it as existing within the system. So, the system will recreate the document and its extracted entities again.

Note: The Entity Extractor server is optional. If your installation does not include this server, the system will not extract the entities defined in your documents.

In some situations, you may have data sources external to Palantir that you cannot import. However, you may still wish to search the data from Palantir. In cases like these, you can convert the data to pXML or DocXML and use Palantir's optional Raptor module to search these data sources. Discussion of Raptor and its features are beyond the scope of this guide, but you should keep it in mind when planning your import/export needs.

Palantir Installation Resources for XML Formats

Your Palantir installation comes with resources that support the input and export of pXML and DocXML files. These include optional servers, services, command line utilities, and user interface features.

Optional Servers and Services

The Entity Extraction Server is optional. It supports automatic extraction of entities from documents. When this server is not installed, you can still import DocXML files and their associated entities.

The Authentication Server is an optional server that supports the use of LDAP to authenticate users. This server allows you to add a number of authentication sources on top of the Palantir internal authentication system. The configuration of this server supports the specification of access control in your XML documents.

Finally, you can use Palantir's open APIs to build additional services that support your users in importing data from external sources. For example, Palantir ships a default entity extraction service for Inxight SmartDiscovery servers. You can build a similar service for others servers.

Command Line Tools

Within your Palantir `PALANTIR_INSTALLATION/server/bin/unix` directory you will find the following commands for working with XML files:

Name	Description
`validatePXML.sh`	Validates pXML files.
`metsToDocXML.sh`	Converts METS XML to DocXML format.
`textResourceEditor.sh`	Edits Palantir DocXML and METS schema Maps.
`docXMLCrawl.sh`	Imports a directory of DocXML files into the Palantir system.
`authSourceConfiguration.sh`	Manages authentication sources.

If you are using a QuickStart installation, refer to the *Palantir Administrator's Guide* for the list of corresponding commands on Windows. You should also refer to the same guide for instructions on using the commands listed above.

User Interface Commands

From within the Palantir Workspace you can use the **Data Import** command to import one or more files into Palantir. This is an easy way to import a single XML file when you are testing a pXML or DocXML form. You can also select one or more objects and choose **Export Selected** to output pXML.

The **Extract Entities** button on the Browser extracts entities from a document that is already in Palantir. If the document already contains extracted entities, the system will not attempt to extract them again.

3 Importing Data with pXML

This chapter describes how to work with the pXML format. You will learn the format's structure and how to use it effectively. You should read this chapter if you want to understand how to construct a pXML document. This chapter contains the following:

- Overview of the Format
- Constructing Objects
- The aclSet and Access Control on Import
- Defining Data Sources

If you are interested in using DocXML, please see Chapter 3, *Importing Data with pXML*.

Overview of the Format

The pXML format allows you to import data into Palantir from an external data source. You can import data from a database or from files in a file system. This section introduces you to the essential structures in pXML and contains the following:

- The Main Elements in pXML
- Walkthrough of a Simple XML

The Main Elements in pXML

A pXML document has a single `graph` element that contains these child elements:

`aclSet`	Details access controls that the pXML references.
`dataSourceSet`	Lists the data sources that the pXML references. Some examples of data sources include Excel spreadsheets, PDF files, or email.
`objectSet`	Contains the objects defined in the pXML document. There is an entry for each object instance.
`linkSet`	Contains a set of link (relationship) elements that apply between object instances.

The elements in the graph are order dependent. You must specify them in the order in which they are listed above. You can specify one, some, or all of the these elements but you must maintain the relative order.

Walkthrough of a Simple XML

This section guides you through a simple example of an pXML. This pXML represents a `graph` that contains a single `dataSource` element:

```
<?xml version='1.0' encoding='UTF-8'?>
<palantir xmlns="http://www.palantirtech.com/pg/schema/import/">
<graph>
  <dataSourceSet>
    <dataSource id='PT_DATASOURCE1' type='com.palantir.datasource.user' >
      <name>Manually Entered Data</name>
      <description>Data entered manually through Palantir
                   Workspace</description>
    </dataSource>
  </dataSourceSet>
```

This pXML defines a single object, `PT_OBJECT2` of type `com.palantir.object.Bombing` which is an event.

```
  <objectSet>
    <object id='PT_OBJECT2' type='com.palantir.object.Bombing'>
    <propertySet>
      <property id='PT_PROPERTY3'
        type='com.palantir.property.IntrinsicTitle'
        linkType='com.palantir.link.Title' role='com.palantir.role.none'
        keywordDisabled='false' >
        <propertyValue>
            <propertyComponent type='TITLE' >
```

```
        <propertyData>Blammo</propertyData>
      </propertyComponent>
      <propertyComponent type='PRIORITY' >
          <propertyData>1000000000</propertyData>
      </propertyComponent>
    </propertyValue>
```

The `dataSourceRecord` on the property refers back to the `dataSource` defined earlier in the file.

```
    <dataSourceRecordSet>
      <dataSourceRecord dataSource='PT_DATASOURCE1'
                importKey='8515422156009064155' recordLocator='0' >
      </dataSourceRecord>
    </dataSourceRecordSet>
  </property>
  <property id='PT_PROPERTY4' type='com.palantir.property.EventTitle'
    linkType='com.palantir.link.Simple' role='com.palantir.role.none'
    keywordDisabled='false' >
    <propertyValue>
        <propertyData>Blammo</propertyData>
    </propertyValue>
    <dataSourceRecordSet>
        <dataSourceRecord dataSource='PT_DATASOURCE1'
                importKey='8515422156009064155' recordLocator='0' >
        </dataSourceRecord>
    </dataSourceRecordSet>
  </property>
```

pXML supports the specification of time values on specific properties using the `timestamp` or `timeInterval` elements. You can set time specific to an `object` using a `propertyTimeInterval` element. See Object Labels and Time on page 19 for information on specifying time values on objects and properties.

```
  <property id='PT_PROPERTY5'
            type='com.palantir.property.TimeInterval'
            linkType='com.palantir.link.TimeInterval'
            role='com.palantir.role.none' keywordDisabled='false' >
      <timestamp timestamp='2008-07-25T12:07:14.337-07:00' />
      <propertyValue>
          <propertyTimeInterval
              timeStart='2008-07-25T12:07:14.337-07:00'
              timeEnd='2008-07-25T12:07:14.337-07:00' />
      </propertyValue>
      <dataSourceRecordSet>
        <dataSourceRecord dataSource='PT_DATASOURCE1'
                  importKey='1370165967854021746' recordLocator='0' >
        </dataSourceRecord>
        <dataSourceRecord dataSource='PT_DATASOURCE1'
        importKey='8577310725570520937' recordLocator='0' >
        </dataSourceRecord>
      </dataSourceRecordSet>
```

```
      </property>
    </propertySet>
  </object>
</objectSet>
```

This particular pXML does not contain any relationships so the `linkSet` element is empty.

```
    <linkSet>
    </linkSet>
  </graph>
</palantir>
```

Constructing Objects

This section discusses some general principles you need to understand when constructing a pXML document. The following topics appear in this section:

- Working with ID Fields
- Simple and Composite Properties
- Object Labels and Time

Working with ID Fields

IDs within the pXML are internally referential. Simply put, this means within a pXML you must refer to an ID that exists previously within the document. The exception is the `aclReference` element, this can reference an ID defined in Palantir's ACL schema and not in the pXML. (You will learn more about the `aclReference` later.)

Palantir assigns its own IDs to the objects. If you are doing a one-time import of data, the IDs in your pXML only serve to help you construct the pXML.

If you are constructing pXML for use by Palantir's Raptor search module, then the IDs serve an additional purpose. Raptor uses this ID to watch external data sources for changes. When the data associated with that external ID changes, Raptor ensures the repository representation reflects the change.

Simple and Composite Properties

An object can have simple or composite properties. Composite properties, as you might expect, are composed of multiple components. You can determine which category a property falls into, simple or composite, by examining the property's **Base type** field in the Ontology Manager.

Simple properties are of type String, Number, or Date. In the case of a simple property, you place a `propertyData` element directly within a `propertyValue` element. For example, as you might expect, `com.palantir.property.Age` is a simple property with a base type of `Number`. Within your XML you an specify an `age` property as follows:

```
<property id='PT_PROPERTY7' type='com.palantir.property.Age'
          linkType='com.palantir.link.Simple'
role='com.palantir.role.none'
          keywordDisabled='false' >
  <propertyValue>
    <propertyData>32</propertyData>
  </propertyValue>
  ...
</property>
```

A composite property is composed of multiple `propertyComponent` elements. Use the Ontology Manager to list the components of a composite property. For example, the `name` property is a composite property:

```
<property id='PT_PROPERTY4' type='com.palantir.property.Name'
          linkType='com.palantir.link.Simple'
role='com.palantir.role.none'
          keywordDisabled='false' >
  <propertyValue>
      <propertyComponent type='FIRST_NAME' >
        <propertyData>the</propertyData>
      </propertyComponent>
      <propertyComponent type='LAST_NAME' >
        <propertyData>dude</propertyData>
      </propertyComponent>
  </propertyValue>
```

Specify each component of the composite property only once. However, you need not specify all components, you can specify a subset. For instance, the example above does not provide the `MIDDLE_NAME` component in the `com.palantir.property.Name` definition.

If you want the system to import a property's value using the parser, you can use the `propertyRawValue` element in place of the `propertyData` element. You can see the parsers applied to a property using the Ontology Manager. You can use this element with simple or composite properties.

Object Labels and Time

You can define two special composite properties —the `com.palantir.property.IntrinsicTitle` and the `com.palantir.property.TimeInterval`. To construct these properties, you must set the `property` element's `type` attribute to one of these values.

In the majority of cases, you should use the `title` element on the `object` element. However, you may have the case of an entity with two properties (for example, `name` and `location`) from two different data sources. Suppose the `name` property has one ACL and the `location` another. In this case, you may want to display different titles to the members of the different ACLs. You can use the `IntrinsicTitle` property to do this.

The `IntrinsicTitle` is a composite property that defines the object title or label that the system displays to a user. If you use this property type, then you must set the property's `linkType` attribute to `com.palantir.link.Title`. You must also include a `propertyValue` element with components of `TITLE` and `PRIORITY`. This ensures the title displays correctly; otherwise the title appears as an object property and not its label.

You can specify multiple `IntrinsicTitle` properties as well as the `title` property. Palantir will display the one with the highest priority. The data importer determines the priority automatically on import. The `title` priority always has the highest priority.

The `com.palantir.property.TimeInterval` property type specifies a time interval for the entire object, like the duration of a phone call event. Typically, you specify time intervals on event and document objects. An example event might be a prison sentence from April 3rd, 2001 through June 5th, 2008. This would appear in XML as:

```
<property id='PT_PROPERTY5' type='com.palantir.property.TimeInterval'
          linkType='com.palantir.link.TimeInterval'
          role='com.palantir.role.none'
          keywordDisabled='false' >
  <propertyValue>
    <propertyTimeInterval timeStart='2001-04-03T12:09:52.325-08:00'
                          timeEnd='2008-06-25T12:09:52.325-07:00' />
  </propertyValue>
```

The `propertyValue` must also contain a `propertyTimeInterval` child element. Finally, for the value to display correctly, also set the property's `linkType` attribute to `com.palantir.link.TimeInterval`.

The `timeInterval` or `timestamp` element on a `property` specifies a time value for that specific property. For example, a person has an cell phone property. That phone may have been in effect for only a month. The following snippet from an XML illustrates how to construct this:

```
<property id='PT_PROPERTY6' type='com.palantir.property.Phone'
          clinkType='com.palantir.link.Simple'
          role='com.palantir.role.none'
```

```
        keywordDisabled='false' >
  <timeInterval timeStart='2008-07-01T11:56:00.000-07:00'
                timeEnd='2008-08-21T11:56:00.000-07:00' />
  <propertyValue>
      <propertyComponent type='COUNTRY_CODE' >
        <propertyData>1</propertyData>
      </propertyComponent>
      <propertyComponent type='AREA_CODE' >
        <propertyData>650</propertyData>
      </propertyComponent>
      <propertyComponent type='PHONE_NUMBER' >
        <propertyData>4488585</propertyData>
      </propertyComponent>
  </propertyValue>
...
</property>
```

The aclSet and Access Control on Import

This section contains topics and tips for specifying access control on the data you are importing. The following topics appear here:

- Writing an aclSpecification
- Specify a New ACL
- Reference an ACL
- Creating a Default ACL
- How the Data Importer Resolves ACLs

Writing an aclSpecification

The `aclSpecification` element represents access control lists (ACLs) within your data source. When creating a pXML you can include access control specifications or not. The following elements can take on access control values in the XML:

- `dataSourceRecord`
- `property`
- `media`
- `note`
- `dataSource`

- link

You define access control specifications in the aclSet and then reference these ACLs from an element.

The aclSet element appears as the first child within a graph element. The aclSet contains a list of one or more acl elements and at most one defaultAcl tag. The following example illustrates the possible constructions within an aclSet:

```
<aclSet>
  <defaultAcl aclId="ACL_1">
      <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                        externalId="Everyone"
                        permissions="owner"/>
      <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                        externalId="Administrators"
                        permissions="write"/>
  </defaultAcl>
  <acl aclId="ACL_2">
    <aclReference aclId="115"/>
  </acl>
  <acl aclId="ACL_3">
    <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                      externalId="theTestGroup3"
                    permissions="write" />
  </acl>
</aclSet>
```

Within an acl or defaultAcl element, there must be at least one group from the investigation group set. If you forget to include such a group, the Data Importer will reject your XML as invalid.

Specify a New ACL

ACL_3 below illustrates how to create a totally new ACL. You use the aclSpecification element to define a new ACL consisting of a group-permissions pair:

```
<acl aclId="ACL_3">
  <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                    externalId="theTestGroup3"
                    permissions="write" />
</acl>
```

The externalId attribute is required. You use it to reference the group ID.

The aclSpecification has an optional authSourceTag attribute that identifies where the ACL originates. This value is case-sensitive.

You must specify an authentication source known to Palantir's dispatch server. Palantir's own internal authentication service or you can use an alternate source known to the dispatcher. The Palantir administrator configures the known authentication sources. You can use the `authSourceConfigurator.sh` command to list the known authentication sources.

If you do not specify the `authSourceTag`, the system will attempt to locate the `externalId` across all known authentication sources. The Data Importer expects to locate only one matching authentication source. If it encounters 0 or more than 1 source, it reports an error and exits.

The `permission` attribute defines the access level to allow for the group. You can set this attribute to one of the following values (in increasing order):

`discovery`	Users can discover information exists during a search, but not read, change, or delete the information.
`read`	Users can discover and read information, but not change or delete it.
`write`	Users can discover, read, change, and delete information.
`owner`	Users can discover, read, change, and delete information, as well as change the permissions of the associated resource.

A higher permission level implies the lower levels. For example, giving a user `write` permissions automatically gives them `read` and `discovery` as well.

Reference an ACL

Your `aclSet` can reference an existing ACL by referencing its numeric ID in the `aclReference` element:

```
<acl aclId="ACL_2">
  <aclReference aclId="115"/>
</acl>
```

The `aclId` on the `aclReference` must correspond to an ID for an ACL already defined in Palantir. Within the pXML, elements must reference this ACL using the `aclId` on the `acl` element.

Creating a Default ACL

If you define a `defaultAcl` element, it must be the first ACL in the `aclSet`. You can specify a new ACL for this default or refer to an existing ACL. The example below illustrates a new ACL construction:

```
<defaultAcl aclId="ACL_1">
    <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                      externalId="Everyone"
                      permissions="owner"/>
    <aclSpecification authSourceTag="COM.PALANTIR.AUTH.INTRINSIC"
                      externalId="Administrators"
                      permissions="write"/>
</defaultAcl>
```

All elements that take an optional `aclId` attribute will use the default ACL provided there is no ACL specified on the individual elements.

How the Data Importer Resolves ACLs

The system applies ACLs in a hierarchical manner through the graph. The following hierarchy illustrates each point on the graph where a user can set an `aclId` attribute.

```
<graph>
  <aclSet>
    <defaultAcl>
    <acl>
  <dataSourceSet>
    <dataSource>
  <objectSet>
        <dataSourceRecord>
        <propertySet>
          <property>
            <dataSourceRecordSet>
              <dataSourceRecord>
        <mediaSet>
          <media>
            <dataSourceRecordSet>
              <dataSourceRecord>
        <noteSet>
          <note>
            <dataSourceRecordSet>
              <dataSourceRecord>
  <linkSet>
      <link>
        <dataSourceRecordSet>
          <dataSourceRecord>
```

The value of the `aclId` attribute on these elements must correspond to one of the `aclId` values on a `acl` or `defautlAcl` element.

Note: The `aclId` element of an `aclReference` element is a reference to the numeric ID of an already existing Palantir ACL and has no correspondence to any other ID or ID reference in the pXML document.

First the system looks for an `aclId` attribute on the `property`, `media`, `note`, or `link` element. If it the `aclId` attribute exists, the system applies it.

If the attribute does not exist on the element, the system looks for an `aclId` on the `dataSourceRecord` element and applies that if it exists. If it does not exist, the system reviews the `dataSource` element's `aclId` attribute and uses that. If the `dataSource` has no `aclId`, the system uses the `defaultAcl`.

If you do not specify a `defaultACL` on your graph, the Data Importer uses the creation ACL for the current investigation or the ACL defined for the import session.

Defining Data Sources

A Palantir data source supplies information that is imported either by an analyst or an administrator. There are three basic types of data sources.

structured	A database or any tabular delimited file such as .csv file.
semi-structured	An individual email or an email server.
unstructured	A document which may or may not be in pXML format.

You can think of a data source as the means for tracking the pedigree or lineage of external data. This section discusses how the pXML supports the specification of data sources.

The Data Source Record Hierarchy

Within the pXML, you specify a set of data sources. Then, you refer back to these sources from the object graph using the ID:

```
<dataSourceSet>
  <dataSource id=”10”>
...
```

```
<objectSet>
    <object>
      ...
      <dataSourceRecord dataSource=”10”>
      <note>
        <dataSourceRecordSet>
            <dataSourceRecord dataSource=”22” ...>
...
<linkSet>
    <link>
    ...
      <dataSourceRecord dataSource=”23”>
...
```

The objects and links in your graph use `dataSourceRecord` elements to reference back to a data source. Further down the hierarchy, each `note`, `media`, and `property` element associated with an object also has a data source. Each data source in the hierarchy must reference a data source in the graph's set.

Import Keys and String Position Locators

Every data source record must have a unique identifier. For a structured data source, an import key is that identifier. The key points to the specific record the data comes from such as a database row or an index.

For an unstructured data source, you should specify a 0-length import key and then specify a unique string position locator (SPL). The `stringPositionLocator` element details the offsets in the unstructured source that pinpoint a data's location. For example:

```
<dataSourceRecord dataSource="PT_DATASOURCE18" importKey="">
    <stringPositionLocator startPosition="843"
                           endPosition="863"
                           sentenceNumber="28"
                           paragraphNumber="33"/>
  ...
</dataSourceRecord>
```

It is an error to provide both a `stringPositionLocator` and a non-zero-length `importKey`.

Typing Data Sources

When you define a data source, you specify a `type` value. The `type` value determines which icon the system uses when displaying the component in the Workspace. The `type` value also supplies some information during programmatic access to the data source. The types you will find yourself using most frequently are the following:

`com.palantir.datasource.filetext`	Generic file system document (unstructured documents).
`com.palantir.datasource.dbtable`	A generic DB tablespace.
`com.palantir.datasource.filedocument`	Structured Text File

See the `dataSourceRecord` element on page 43 for a complete list of the possible data sources.

If you specify an unstructured `type`, your `dataSource` element must contain a `primaryObject` child element if you want to have the data available in the document viewer. The data source `title` appears in the Palantir user interface. For unstructured data sources, the system uses the `title` from the primary object.

The Data Importer does not attempt to perform data source resolution. So, if it encounters two data sources with the same name, the importer does not merge them into one.

4 Working with DocXML

This chapter describes how to work with the DocXML format. You will learn the format's structure and how to use it effectively. The following topics appear in this chapter:

- Overview of the DocXML Format
- Working with Metadata and Content
- Working with Objects and Relationships

Overview of the DocXML Format

This section describes the DocXML format, its functionality, and the major sections it contains. The DocXML format is a smaller set of XML components than you will find in pXML. In come cases, DocXML contains the same tags as pXML but uses them in a slightly different way. So, even if you are very familiar with pXML, you will benefit from this discussion.

Functionality of the DocXML Format

Each DocXML file describes a single document. The format is designed to capture not only the document content but the information within the document you wish to extract into Palantir. Using this format you can import into Palantir:

- the document content
- metadata about the document such as data sources and access control
- entities referenced in the document

- events referenced in the document
- relationships between the entities and events the document references

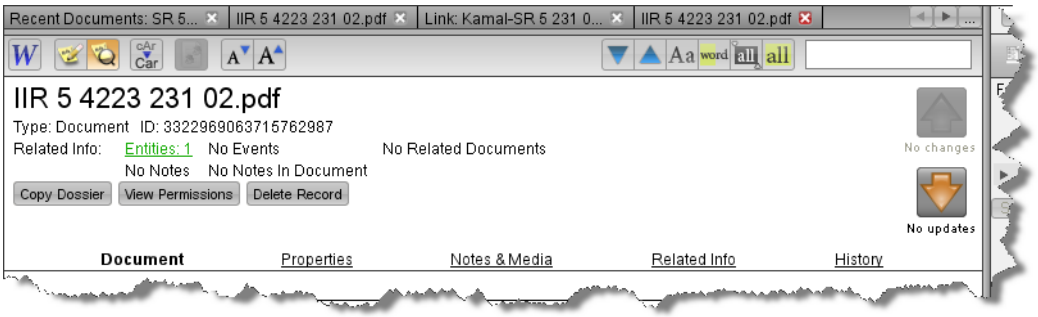
Within DocXML, you specify the length in characters of entity and event references and specify their locations with offsets. On import, the system uses these values when determining an entity's location in the document. Once the document is in Palantir, the Browser displays the location or tags for each entity you referenced.

Major Elements in DocXML

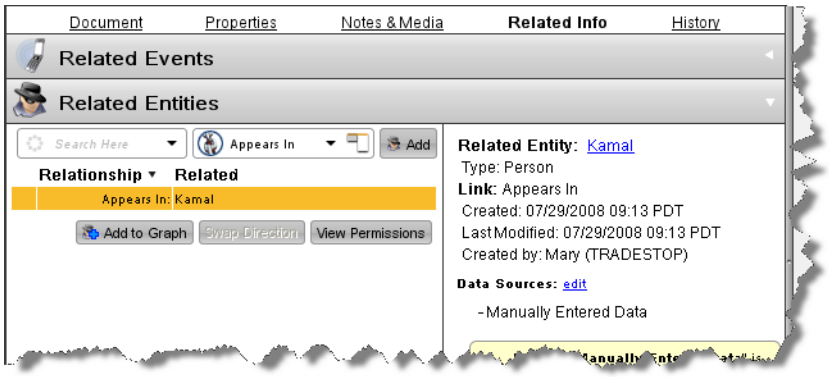
Each DocXML contains a single `document` element that is required. Within the `document` is a single `documentMetaData` element that is also required. Optionally, you can define any number of `objectData` elements or none at all. Finally, the document element contains a `content` element with the encoded document.

You use the `documentMetaData` element to define information associated with the document it represents. There are some standard child elements such as the title, data source, encoding, and classification. The `title` appears as the document's label in the Workspace Graph:

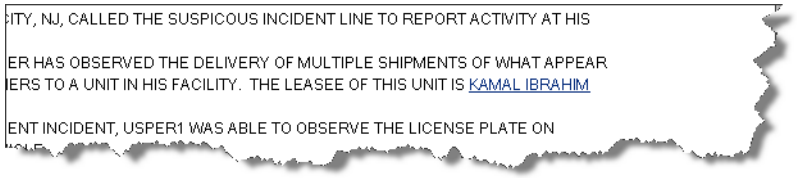
Additionally, you can associate arbitrary metadata with your document using the `documentProperties` element. As you would expect, this element lists the properties that apply to the document. These properties appear in the Workspace in the **Properties** section:



The `objectData` element encapsulates the objects referenced in the document using the `objectSet` element. The relationships among these objects are described through the `relationshipSet` element. The `objectSet` element is required if you are defining `objectData`. The `relationshipSet` element is not. The information from these sections is visible in the Browser after you import the document:



For each object element you identify in a set, you have the option of specifying a `textReference` that pinpoints the location in the document where a reference occurs:



If you have installed the entity extractor, then, when you import the DocXML, the system extracts the objects or entities you defined in the DocXML into Palantir.

A Simple Document Example

This section introduces you to the DocXML format through the means of a simple example. Consider the following text document:

```
SAMPLE CLASSIFICATION
----------------------------------------------------------------
DR011840506 NRS 30-040-678 134-3244-272 1557 05OCT07 CIA 57062
----------------------------------------------------------------
TO: CIA/CTC FROM: CAIRO STATION
ACTION: CIA/CTC

1. ACTION: FOR INFO ONLY.
```

```
2. Agent CT BLUE Attended AL-MUJHA Charity Event in Maddi, outside of
Cairo. Several US nationals were in attendance. CT BLUE gathered three (3)
business cards from attending nationals.

3. BUSINESS CARD 1: Name: Mohd Fikri Phone: 925-950-4901 Email:
MFIKRI@YAHOO.COM Address: 19438 SE 14TH STREET, LIVERMORE, CA 94550-4763

4. BUSINESS CARD 2: Name: OMAR ATARJI Phone: 415-590-7446 Email:
OMAR.ATARJI@HOTMAIL.COM Address:
13406 157TH STREET CT. E., SAN FRANCISCO, CA 94110

5. BUSINESS CARD 3: Name: STEVEN AL SHAMMARI Phone: 650-519-3398 Email:
AL_SHAMMARI7@AOL.COM Address: 1732 EAST 30TH STREET, LARKSPUR, CA 94939

SAMPLE CLASSIFICATION
```

A DocXML representation of this document would be as follows:

```
<palantir xmlns="http://www.palantirtech.com/pg/schema/docxml/">
  <document>
  <documentMetaData>
    <title>Cable123: CT Blue on AL-MUJHA Activity in Maddi</title>
    <source>file:/C:\myfile.xml</source>
    <contentType>text/plain</contentType>
    <encoding>UTF-8</encoding>
    <timestamp>2007-12-11T17:50:40-08:00</timestamp>
    <classification>UC</classification>
    <sentenceBreaks offsets="0, 24, 253, 287, 304, 331, 407, 448,
                    518, 657, 808, 954"/>
    <paragraphBreaks offsets="0, 24, 253, 287, 304, 331, 518, 657, 808,
                     954"/>
  </documentMetaData>
<content>IFNBTVBMRSBDTEFTU0lGSUNBVElPTgoKIC0tLS0tLS0tLS0tLS0tLS0tLS0t
LS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tL
S0tLSBEUjAxMTg0MDUwNiBOUlMgMzAtMDQwLTY3OCAxMzQtMzI0NC0yNzIgMTU1NyAwNU9DVD
A3IENJQSA1NzA2MiAtLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0KClRPOiBDSUEvQ1RDICBG
Uk9NOiBDQUlSTyBTVEFUSU9OCgpBQ1RJT046IENJQS9DVEMKCjEuIEFDVElPTjogRk9SIElOR
k8gT05MWS4KCjIuIEFnZW50IENUIEJMVUUgQXR0ZW5kZWQgQUwtTVVKSEEgQ2hhcml0eSBFdm
VudCBpbiBNYWRkaSwgb3V0c2lkZSBvZiBDYWlyby4gU2V2ZXJhbCBVUyBuYXRpb25hbHMgd2V
yZSBpbiBhdHRlbmRhbmNlLiBDVCBCTFVFIGdhdGhlcmVkIHRocmVlICgzKSBidXNpbmVzcyBj
YXJkcyBmcm9tIGF0dGVuZGluZyBuYXRpb25hbHMuCgozLiBCVVNJTkVTUyBDQVJEIDE6ICBOY
W1lOiBNb2hkIEZpa3JpIFBob25lOiA5MjUtOTUwLTQ5MDEgRW1haWw6IE1GSUtSSUBZQUhPTy
5DT00gQWRkcmVzczogMTk0MzggU0UgMTRUSCBTVFJFRVQsIExJVkVSTU9SRSwgQ0EgOTQ1NTA
tNDc2MwoKNC4gQlVTU5FU1MgQ0FSRCAyOiAgTmFtZTogT01BUiBBVEFSSkkgUGhvbmU6IDQx
NS01OTAtNzQ0NiBFbWFpbDogT01BUi5BVEFSSklASE9UTUFJTC5DT00gQWRkcmVzczogMTM0M
DYgMTU3VEggU1RSRUVUIENULiBFLiwgU0FOIEZSQU5DSVNDTywgQ0EgOTQxMTAKCjUuIEJVU0
lORVNTIENBUkQgMzogIE5hbWU6IFNURVZFTiBBTCBTSEFNTUFSSSBQaG9uZTogNjUwLTUxOS0
zMzk4IEVtYWlsOiBBTF9TSEFNTUFSSTdAQU9MLkNPTSBBZGRyZXNzOiAxNzMyIEVBU1QgMzBU
SCBTVFJFRVQsIExBUktTUFVSLCBDQSA5NDkzOQoKU0FNUExFIENMQVNTSUZJ
Q0FUSU9OCgoKCiAgIA==</content>
    <objectdata>
```

```
      <objectSetMetaData externalID="My DocXML Source" />
        <objectSet>
          <object type="Meeting" objectID="OBJECT1">
            <textReference characterOffset="366" characterLength="13"
                           propertyType="EventName">
                <value>Charity Event</value>
            </textReference>
            <textReference characterOffset="383" characterLength="5"
                           propertyType="Location">
                <value>Maddi</value>
            </textReference>
          </object>
          <object type="Person" objectID="OBJECT2">
              <textReference characterOffset="684" characterLength="11"
                             propertyType="PersonName">
            <value>Omar Atarji</value>
            </ textReference>
          </object>
          <object type="Person" objectID="OBJECT3">
            <textReference characterOffset="545" characterLength="10"
                           propertyType="PersonName">
                <value>Mohd Fikri</value>
            </textReference>
            <textReference characterOffset="563" characterLength="12"
                           propertyType="Phone">
                  <value>925-950-4901</value>
            </textReference>
          </object>
        </objectSet>
        <relationshipSet>
          <relationship childRef="OBJECT2" parentRef="OBJECT3"
                        relationType="ManagerOf" />
          <relationship childRef="OBJECT1" parentRef="OBJECT3"
                        relationType="AppearsIn" />
        </relationshipSet>
      </objectdata>
    </document>
</palantir>
```

The `palantir` element encloses all Palantir file formats — both pXML and DocXML. The `xmlns` element defines the specific namespace or format and in this case it is DocXML. All DocXML must begin with these two tags.

The first part of the document definition is the metadata. The importer expects this data in a well-defined sequence. Not all child elements must be present. For example, the `acl` is the last item in the `documentMetaData` element but it is omitted in this example.

It is in the `documentMetaData` that you will define the document's encoding. You must specify an encoding recognized by the Java platform. The `source` tag specifies the URI for the document. In this example, it identifies UTF-8 as the encoding:

```
<palantir xmlns="http://www.palantirtech.com/pg/schema/docxml/">
```

```
  <document>
  <documentMetaData>
    <title>Cable123: CT Blue on AL-MUJHA Activity in Maddi</title>
    <source>file:/C:\myfile.xml</source>
    <contentType>text/plain</contentType>
    <encoding>UTF-8</encoding>
```

The `content` element contains the actual encoded content of the document. This element must hold a Base64 encoded value.

The `objectData` section begins with an `objectSetMetaData` value. The `objectData` section contains all of the objects mentioned in the document. It also details the objects' properties and the relationships to other objects mentioned in the document:

```
    <objectdata>
      <objectSetMetaData externalID="My DocXML Source" />
        <objectSet>
          <object type="Meeting" objectID="OBJECT1">
            <textReference characterOffset="366" characterLength="13"
                          propertyType="EventName">
                <value>Charity Event</value>
            </textReference>
...
```

The `externalID` provides a link back to the DocXML Schema Map. The Schema Map supports the use of multiple DocXML generators by allowing you to use external types in your DocXML files. The Schema Map then "maps" these external types to Palantir types in your ontology. (You will learn more about the Schema Map later.)

This example defines three objects that are identified by their `type` attributes — two `Person` objects and one `Meeting` object. The `objectID` attributes provided here are optional. You need only specify these if you plan to reference the object from a `relationship` element.

In this example, the document contains the event's name. The `textReference` element identifies the section of the document containing the name with offsets. It also identifies the `propertyType` on the event object that this reference represents — `PersonName`. This property is located at position 366, and is 13 characters long.

An individual object may have multiple properties of the same type. For example, a person object may be known by multiple names (such as Bill Clinton and William Jefferson Clinton). The values you use for a reference need not match the text reference format. Take the following:

```
<object type="Person" objectID="OBJECT2">
  <textReference characterOffset="684" characterLength="11"
                propertyType="PersonName">
      <value>Omar Atarji</value>
  </textReference>
</object>
```

This object block specifies a `Person` with a single `PersonName` and a value of `Omar Atarji`. In the original document text Omar's name is capitalized (`OMAR ATARJI`) yet the `value` is not. You would specify this alternative capitalization if you want the property to appear in the Workspace as `Omar Atarji`.

Although all the `Person` types in this example have a `Name` property, it is not required. You can supply any subset of properties you like for an object. For instance, if a text references a phone number but no name, you can create a person object with a phone number property and no name property. This would appear as follows:

```
<object type="Person" objectID="OBJECT3">
  <textReference characterOffset="563" characterLength="12"
                 propertyType="Phone">
         <value>925-950-4901</value>
  </textReference>
</object>
```

The `relationshipSet` element contains all the relationships between objects mentioned in the document:

```
<relationshipSet>
    <relationship parentRef="OBJECT3" childRef="OBJECT2"
                  relationType="ManagerOf" />
    <relationship parentRef="OBJECT3" childRef="OBJECT1"
                  relationType="AppearsIn" />
</relationshipSet
```

The `objectID` values from the `object` elements resolve the objects in this code snippet. In this case, `OBJECT2` refers to Omar Atarji and `OBJECT3` refers to Mohd Fikri.

You can specify a symmetrical relationship or an asymmetrical relationship. You can think of a symmetrical relationship as balanced — neither side of the relationship takes social, power, or other precedence in any way. An asymmetrical relationship is imbalanced in some way — for example a coach to player or employer to employee.

The `managerOf` relationship between the Mohd Fikri and Omar Atarji is an asymmetrical relationship. `OBJECT3` (Mohd) is the parent and `OBJECT3` (Omar) is the child. For symmetrical relationships, it does not matter which object appears in which reference. A symmetrical relationship might be "sibling of" or "graduated with."

The last three tags close out the document.

```
    </objectData>
  </document>
</palantir>
```

When a document is imported, the validator checks to make sure the XML is properly formed.

Working with Metadata and Content

This section discusses specific topics related to working with metadata and document content. The following topics are discussed:

- Importing Different Document Types
- Document MetaData
- Adding Property Elements

Importing Different Document Types

When defining a DocXML file, you can assign a Palantir document type to the document. You do this by supplying a `palantirType` attribute on the document. The following example identifies the document as a `com.palantir.object.MessageTraffic` document:

```
<?xml version="1.0" encoding="utf-8"?>
  <palantir xmlns="http://www.palantirtech.com/pg/schema/docxml/">
    <document palantirType="com.palantir.object.MessageTraffic">
      ...
    </document>
</palantir>
```

The `palantirType` attribute on the document element is optional.

If you define a `palantirType` attribute and specify a document subtype in the Palantir ontology, the data importer maps the DocXML document to that subtype. The data importer maps the document to Palantir's intrinsic type, `com.palantir.object.Document`, when the `palantirType` attribute is:

- absent
- present but specifies a type that does not exist in the Palantir ontology
- present but specifies a type that exists in the Palantir ontology but is not a document subtype

This behavior only occurs when you are importing new DocXML documents. You cannot change the type associated with an existing document.

Document MetaData

The `documentMetaData` element supplies information about your document. The element's children must appear in a specific sequence which is as follows:

- `title`
- `datasourceTitle`
- `source`
- `contentType`
- `encoding`
- `timestamp`
- `classification`
- `documentProperties`
- `sentenceBreaks`
- `paragraphBreaks`
- `acl`

You can omit the optional child elements but you still must maintain this relative order. The following illustrates an example of `documentMetaData` that includes an `acl` element.

```
...
<documentMetaData>
  <title>010657Z SEP 01_2_response.xml</title>
  <source>file:/c:/import_docs/010657z sep 01_2_response.xml</source>
  <contentType>text/plain</contentType>
  <encoding>UTF-8</encoding>
  <timestamp>2005-02-25T09:56:00-08:00</timestamp>
  <acl>
    <aclSpecification externalId="Everyone" permissions="read"/>
    <aclSpecification externalId="Administrators" permissions="write"/>
    <aclSpecification externalId="Backup Users" permissions="owner"/>
  </acl>
</documentMetaData>
...
```

The `title` field sets the title for both the document and its data source. You can specify the optional `dataSourceTitle` element. If you do, it will override the data source name.

The `source` element's value is a URI. The importer uses the value to set the document's `com.palantir.property.Filename` value. If you leave `source` blank, the data importer uses the DocXML file name. If your installation's ontology does not include the `Filename` property, the importer ignores it.

The system uses the `timestamp` elements as the time interval on the document object — it sets the start and end times to the value you supply. The system uses this timestamp when placing the document on the Timeline.

You can specify an optional `documentProperties` element. See Adding Property Elements on page 36 for information on defining properties.

The `sentenceBreaks` and `paragraphBreaks` are both optional. A break is defined as the character offset (0-based) where sentences and paragraphs start. If you define `sentenceBreaks` but not `paragraphBreaks`, the importer generates the missing set.

Once the system identifies the sentence breaks, it merges into these the paragraphs. In this way, one sentence cannot span two paragraphs.

In general, most DocXML does not specify sentence and paragraph breaks. If you leave both of these elements out, the Data Importer applies breaks for you. You should permit this unless your export pipeline that uses them.

Adding Property Elements

Unlike pXML, DocXML does not distinguish between simple and composite properties. You simply supply the property's type and a string value to the importer. The system applies the parser that is appropriate for the phone type:

```
<documentProperties>
  <property palantirType="com.palantir.property.phone"
            value="650-995-8545" />
  <property palantirType="com.palantir.property.publication"
          value="Sudan -- FMA" />
</documentProperties>
```

You can supply either a `name` or a `palantirType` attribute or both to the importer. If you supply both, the system uses the `palantirType`. If you supply simply a `name` attribute, the DocXML schema `propertyMapping` section must map the `name` attribute into a Palantir property type.

You can attach time and geographic values to a property (or to a textReference) see Text References on page 39 for more information specifying these values.

Working with Objects and Relationships

The `objectData` element defines the objects and relationships related to the document. In a sense, this element is the reason DocXML exists. It holds the information that the entity extractor uses to tag your document. This section discusses topics in defining objects and relationships in DocXML:

- Creating an Object
- Relationship Definition
- Text References

Creating an Object

You use `object` elements to represent entities in a document. An `object` takes 0 or more `textReference` child elements. Using `textReference` elements you can pinpoint where in the document an object is referenced. This causes the importer to "tag" your document. After the importer successfully imports the document and extracts its entities, you can use this tag to jump from the document to the entity.

You can specify `objectID` attributes for each `object` in your document. These IDs are internal to the DocXML document. The `objectID` attribute is not required. However, you need them for referencing the object elements from `relationship` elements in DocXML. The importer does not pull or preserve these identifiers in Palantir in any way.

The `textReference` element is optional. There is no limit on the number of text references you can have in a document. This element has the following child elements:

- `value`
- `timestamp` or `timeInterval`
- `gisData`

The child elements are order dependent, for example, you cannot specify `gisData` before the `value` attribute.

You have the option of specifying an Palantir object type for each `object` and a Palantir property type for each `textReference`. Alternatively or in addition, you can specify external object and property types. The following example shows the use of external property types as opposed to Palantir types:

```
<object type="PT_City" title="Addis Ababa">
  <textReference characterOffset="4748" characterLength="11"
      propertyType="PT_City">
    <value>Addis Ababa</value>
  </textReference>
```

```
  <textReference characterOffset="4963" characterLength="11"
                 propertyType="PT_City">
    <value>Addis Ababa</value>
  </textReference>
</object>
```

If you use external types, your installation's DocXML schema must map the external type to the corresponding Palantir type. You can specify both an external type and a Palantir type. If you do, the importer uses the Palantir type.

Relationship Definition

Once you have a set of `object` elements, you can use the `relationship` element to describe any relationships between them. The `relationshipSet` element contains the relationships between the document's `object` elements. A `relationship` definition consists of:

- a reference to a child object
- a reference to a parent object
- the relationship type

To specify the relationship's type, you must either supply a `relationType` attribute or a `palantirRelationType` attribute or both to the importer. If you supply only a external `relationType` attribute, you must ensure the DocXML schema maps the `relationType` attribute into a Palantir property type. If you supply both, the system uses the `palantirRelationType`. If the `palantirRelationType` is invalid, the `relationType` is used.

The following example uses an external link type:

```
<relationshipSet>
  <relationship childRef='OBJECT4' parentRef='OBJECT6'
      relationType="member" />
</relationshipSet>
```

The type is mapped within the DocXML Schema Map:

```
<?xml version="1.0" encoding="utf-8"?>
<mapping xmlns="http://www.palantirtech.com/pg/schema/docxmlmap/">
...
    <typeMapping externalType="member"
                 palantirLinkType="com.palantir.link.MemberOf" />
...
```

This example uses palantir types so does need to use the Schema Map.

```
<relationshipSet>
```

```
  <relationship childRef="OBJECT4" parentRef="OBJECT6"
                palantirRelationType="com.palantir.link.MemberOf"/>
</relationshipSet>
```

Text References

Text references pinpoint the location of an object's reference in a document. For example, consider the references below. The reference to the `Charity Event` is at position 366 and is 13 characters long:

```
<object type="Meeting" objectID="OBJECT1">
  <textReference characterOffset="366" characterLength="13"
                 propertyType="EventName">
    <value>Charity Event</value>
  </textReference>
  <textReference characterOffset="383" characterLength="5"
                 propertyType="Location">
     <value>Maddi</value>
  </textReference>
</object>
```

The value of a `textReference` is always a string. It is best to use as small a string of characters as possible when referencing text.

The `value` element need not be an exact match to the text reference in the source. You might want to do this if the original text is in German, for example, but instead you wish to have an English translation in the property.

Note: Positions are 0-indexed. The first character in the document is at position 0 (zero) not 1(one).

If you specify a Palantir type for the `textReference`, the system will use the parser for that type when importing the corresponding `value`. This is an advantage for composite types, for example, an address. The parser will break this property into its known components.

As with pXML, you can specify geographical data in DocXML. You can associate graphical information systems (GIS) data with a `textReference` element (or a `property` element). The `gisData` contains a `point` that represents a GIS coordinate. A `point` element has the following child elements:

`userElevation`	Specifies the elevation of the specified point.
`userLatLong`	Defines latitudinal and longitudinal data making up a point.

`userMGRS`	Specifies the Military Grid Reference System (MGRS) value for a given point.
`userUTM`	Specifies the Universal Transverse Mercator (UTM) value for a given point.

If you specify all or some of these values, you must supply them in the order listed above. If, for example, you only have a `userMGRS` value for the point, you can supply that.

Alternatively, you can specify a `longitude`, `latitude`, or `elevation` attribute on the `point` itself and none of the child elements. Use whichever values are appropriate for your source data.

DocXML also supports `timestamp` and `timeInterval` elements on `textReference` elements. The syntax for specifying time is identical to pXML. The following example illustrates both the use of time values and `gisData` in DocXML:

```
<object title="Denali Jasper" palantirType="com.palantir.object.Person"
        objectID="OBJECT1">
  <textReference characterOffset='14' characterLength='21'
                 palantirPropertyType='com.palantir.property.Name'>
    <value>Denali Jasper</value>
    <timeInterval timeStart="2008-01-01T01:00:00.000-08:00"
                  timeEnd="2008-01-01T01:15:00.000-08:00">
  </textReference>
  <textReference characterOffset='1095' characterLength='3'
                 palantirPropertyType='com.palantir.property.aka'>
    <value>Den</value>
    <timestamp timestamp="2008-01-01T01:30:00.000-08:00"/>
    <gisData>
      <point latitude="35.5000" longitude="44.4000"/>
    </gisData>
    </textReference>
</object>
```

5 PalantirXMLImportSchema.xsd File

This chapter contains a complete listing of the PalantirXMLImportSchema.xsd document. For documentation on these elements see Chapter 6, *pXML Element Reference*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: PalantirXMLImportSchema.xsd 67963 2008-06-27 22:06:19Z regs $ -
->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          targetNamespace="http://www.palantirtech.com/pg/schema/import/"
          zxmlns:tns="http://www.palantirtech.com/pg/schema/import/"
          elementFormDefault="qualified">

  <xsd:attributeGroup name="databaseMetaData">
    <xsd:attribute name="timeCreated" type="xsd:dateTime"/>
    <xsd:attribute name="lastModified" type="xsd:dateTime"/>
    <xsd:attribute name="createdBy" type="xsd:string"/>
  </xsd:attributeGroup>

  <xsd:complexType name="stringPositionLocator">
    <xsd:attribute name="startPosition" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="endPosition" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="sentenceNumber" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="paragraphNumber" type="xsd:nonNegativeInteger"/>
  </xsd:complexType>

  <xsd:complexType name="dataSourceRecord">
    <xsd:sequence>
      <xsd:element name="stringPositionLocator" minOccurs="0
                   type="tns:stringPositionLocator"/>
    </xsd:sequence>
    <xsd:attribute name="dataSource" type="xsd:IDREF" use="required"/>
    <xsd:attribute name="importKey" type="xsd:string" use="required"/>
    <xsd:attribute name="recordLocator" type="xsd:long"/>
    <xsd:attribute name="aclId" type="xsd:IDREF" use="optional"/>
    <xsd:attributeGroup ref="tns:SecurityAttributes"/>
```

```
</xsd:complexType>

<xsd:complexType name="dataSourceRecordSet">
  <xsd:sequence>
    <xsd:element name="dataSourceRecord" maxOccurs="unbounded"
                 type="tns:dataSourceRecord"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="property">
    <xsd:sequence>
      <xsd:element name="customKeyword"
                  type="tns:customKeyword" minOccurs="0">
      </xsd:element>
      <xsd:choice maxOccurs="1" minOccurs="0">
        <xsd:element name="timestamp" type="tns:timestamp"
                     maxOccurs="1" minOccurs="1"/>
        <xsd:element name="timeInterval" type="tns:timeInterval"
                     maxOccurs="1" minOccurs="1"/>
      </xsd:choice>
      <xsd:element name="gisData" type="tns:gisData" minOccurs="0">
      </xsd:element>
      <xsd:element name="propertyValue" type="tns:propertyValue">\
      </xsd:element>
      <xsd:element name="customKeyword" type="tns:dataSourceRecordSet"
                  minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="type" type="tns:nonEmptyString"
                   use="required"/>
    <xsd:attribute name="linkType" type="tns:nonEmptyString"/>
    <xsd:attribute name="role" type="tns:nonEmptyString"/>
    <xsd:attribute name="keywordDisabled" type="xsd:boolean"/>
    <xsd:attribute name="aclId" type="xsd:IDREF" use="optional"/>
    <xsd:attributeGroup ref="tns:SecurityAttributes"/>
</xsd:complexType>

<xsd:simpleType name="propertyData">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="propertyComponent">
  <xsd:sequence>
    <xsd:element name="propertyData" type="tns:propertyData">
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="type" type="tns:nonEmptyString" use="required"/>
</xsd:complexType>

<xsd:complexType name="propertyValue">
  <xsd:choice minOccurs="0">
    <xsd:element name="propertyData" type="tns:propertyData"/>
```

```
    <xsd:element name="propertyComponent" maxOccurs="unbounded"
                 type="tns:propertyComponent"/>
    <xsd:element name="propertyRawValue" type="tns:propertyRawValue">
   </xsd:element>
    <xsd:element name="propertyTimeInterval"
                type="tns:propertyTimeInterval">
    </xsd:element>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="propertySet">
  <xsd:sequence>
    <xsd:element name="property" type="tns:property"
                maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mediaSet">
  <xsd:sequence>
    <xsd:element name="media" type="tns:media"
                maxOccurs="unbounded"></xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="mediaData">
  <xsd:restriction base="xsd:base64Binary"/>
</xsd:simpleType>

<xsd:complexType name="mediaSet">
    <xsd:sequence>
      <xsd:element name="mediaTitle" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="80"/>
          </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>
    <xsd:element name="mediaDescription" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="4000"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="mediaData" type="tns:mediaData">
    </xsd:element>
    <xsd:element name="dataSourceRecordSet"
          type="tns:dataSourceRecordSet" minOccurs="0"></xsd:element>
  </xsd:sequence>
  <xsd:attribute name="mediaType" use="required"
                type="tns:nonEmptyString">
  </xsd:attribute>
```

```
  <xsd:attribute name="linkType" type="tns:nonEmptyString"
                 use="required"/>
  <xsd:attribute name="mimeType" type="tns:nonEmptyString"
                 use="optional"/>
  <xsd:attribute name="filename" type="xsd:string"/>
  <xsd:attribute name="aclId" type="xsd:IDREF" use="optional"/>
  <xsd:attributeGroup ref="tns:SecurityAttributes"/>
</xsd:complexType>

<xsd:simpleType name="noteData">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="note">
  <xsd:sequence>
    <xsd:element name="noteTitle" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="80"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="noteData">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string"/>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="dataSourceRecordSet"
                 type="tns:dataSourceRecordSet"
                  minOccurs="0"></xsd:element>
  </xsd:sequence>
  <xsd:attribute name="linkType" type="tns:nonEmptyString"/>
  <xsd:attribute name="aclId" type="xsd:IDREF" use="optional"/>
  <xsd:attributeGroup ref="tns:SecurityAttributes"/>
</xsd:complexType>

<xsd:complexType name="object">
  <xsd:sequence>
    <xsd:element name="title" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="80"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="description" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="4000"/>
        </xsd:restriction>
      </xsd:simpleType>
      </xsd:element>
```

```
            <xsd:element name="dataSourceRecord"
                         type="tns:dataSourceRecord”
                         minOccurs="0"></xsd:element>
            <xsd:element name="propertySet" type="tns:propertySet"
                        minOccurs="0">
            </xsd:element>
            <xsd:element name="mediaSet" type="tns:mediaSet" minOccurs="0">
            </xsd:element>
            <xsd:element name="noteSet" type="tns:noteSet" minOccurs="0">
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="required"/>
        <xsd:attribute name="type" type="tns:nonEmptyString" use="required"/>
        <xsd:attribute name="baseType" type="xsd:anyURI"/>
        <xsd:attribute name="timeStart" type="xsd:dateTime"/>
        <xsd:attribute name="timeEnd" type="xsd:dateTime"/>
        <xsd:attribute name="groupFlag" type="xsd:boolean" use="optional"
                      default="false"/>
    </xsd:complexType>

    <xsd:complexType name="noteSet">
      <xsd:sequence>
        <xsd:element name="note" type="tns:note" maxOccurs="unbounded"></
xsd:element>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="objectSet">
      <xsd:sequence>
        <xsd:element name="object" type="tns:object"
                     maxOccurs="unbounded"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="dataSource">
      <xsd:sequence>
        <xsd:element name="name">
          <xsd:simpleType>
            <xsd:restriction base="tns:nonEmptyString">
              <xsd:maxLength value="80"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="description" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:maxLength value="1000"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="primaryObject" type="tns:primaryObject"
                     minOccurs="0">
```

```
    </xsd:element>
  </xsd:sequence>
  <xsd:attributeGroup ref="tns:SecurityAttributes"/>
  <xsd:attribute name="type" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="id" type="xsd:ID" use="required"/>
  <xsd:attribute name="aclId" type="xsd:IDREF"/>
</xsd:complexType>

<xsd:complexType name="dataSourceSet">
  <xsd:sequence>
    <xsd:element name="dataSource" type="tns:dataSource"
                 maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="link">
  <xsd:sequence>
    <xsd:element name="dataSourceRecordSet"
                type="tns:dataSourceRecordSet" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="parentRef" type="xsd:IDREF" use="required"/>
  <xsd:attribute name="childRef" type="xsd:IDREF" use="required"/>
  <xsd:attribute name="type" type="xsd:string" use="required"/>
  <xsd:attribute name="role" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="id" type="xsd:ID"/>
  <xsd:attribute name="aclId" type="xsd:IDREF" use="optional"/>
</xsd:complexType>

<xsd:complexType name="linkSet">
  <xsd:sequence minOccurs="0">
    <xsd:element name="link" type="tns:link" maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="graph">
  <xsd:sequence>
    <xsd:element name="aclSet" type="tns:aclSet" minOccurs="0">
    </xsd:element>
    <xsd:element name="dataSourceSet" type="tns:dataSourceSet"
                minOccurs="0">
    </xsd:element>
    <xsd:element name="objectSet" type="tns:objectSet"></xsd:element>
    <xsd:element name="linkSet" type="tns:linkSet" minOccurs="0">
    </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="palantir">
  <xsd:sequence>
    <xsd:element name="graph" type="tns:graph"></xsd:element>
```

```
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="palantir" type="tns:palantir">
</xsd:element>
<xsd:simpleType name="negativeOneLong">
  <xsd:restriction base="xsd:long">
    <xsd:maxInclusive value="-1"/>
    <xsd:minInclusive value="-1"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="nonNegativeLong">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="primaryObject">
  <xsd:attribute name="objectRef" type="xsd:IDREF" use="required"/>
</xsd:complexType>

<xsd:complexType name="propertyRawValue">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string"/>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="customKeyword">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="propertyTimeInterval">
  <xsd:attribute name="timeStart" type="xsd:dateTime"/>
  <xsd:attribute name="timeEnd" type="xsd:dateTime"/>
</xsd:complexType>

<xsd:complexType name="aclSet">
  <xsd:choice>
    <xsd:sequence>
      <xsd:element name="defaultAcl" type="tns:acl"></xsd:element>
      <xsd:element name="acl" type="tns:acl" minOccurs="0"
                  maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
    <xsd:element name="acl" type="tns:acl" maxOccurs="unbounded">
    </xsd:element>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="acl">
  <xsd:sequence>
    <xsd:choice>
```

```
      <xsd:element name="aclReference" type="tns:aclReference">
      </xsd:element>
      <xsd:element name="aclSpecification"
                   type="tns:aclSpecification"
                   maxOccurs="unbounded"></xsd:element>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="aclId" type="xsd:ID" use="required"/>
</xsd:complexType>

<xsd:complexType name="aclReference">
  <xsd:attribute name="aclId" type="xsd:long"/>
</xsd:complexType>

<xsd:complexType name="aclSpecification">
  <xsd:sequence/>
  <xsd:attribute name="authSourceTag" type="tns:nonEmptyString"
                 use="optional">
  </xsd:attribute>
  <xsd:attribute name="externalId" type="tns:nonEmptyString"
                 use="required">
  </xsd:attribute>
  <xsd:attribute name="permissions" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="discovery"/>
        <xsd:enumeration value="read"/>
        <xsd:enumeration value="write"/>
        <xsd:enumeration value="owner"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<xsd:simpleType name="nonEmptyString">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:whiteSpace value="preserve"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="description">
  <xsd:sequence>
    <xsd:choice maxOccurs="1" minOccurs="0">
      <xsd:element name="point" type="tns:point" maxOccurs="1"
                   minOccurs="1">
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="point">
```

```
    <xsd:sequence minOccurs="0" maxOccurs="1">
      <xsd:element name="userLatLong" type="tns:userLatLong"
                  maxOccurs="1" minOccurs="0">
      </xsd:element>
      <xsd:element name="userUTM" type="tns:userUTM" maxOccurs="1"
                minOccurs="0">
      </xsd:element>
      <xsd:element name="userMGRS" type="tns:userMGRS" maxOccurs="1"
            minOccurs="0">
      </xsd:element>
      <xsd:element name="userElevation" type="tns:userElevation"
                   maxOccurs="1" minOccurs="0">
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="latitude" type="xsd:double" use="required">
    </xsd:attribute>
    <xsd:attribute name="longitude" type="xsd:double" use="required">
    </xsd:attribute>
    <xsd:attribute name="elevation" type="xsd:double"></xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userLatLong">
    <xsd:attribute name="latitude" type="xsd:string" use="required"></
xsd:attribute>
    <xsd:attribute name="longitude" type="xsd:string" use="required"></
xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userElevation">
    <xsd:attribute name="elevation" type="xsd:string" use="required"></
xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userUTM">
    <xsd:attribute name="utmZone" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="utmNorthing" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="utmEasting" type="xsd:string" use="required">
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userMGRS">
    <xsd:attribute name="mgrs" type="xsd:string" use="required">
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="timeInterval">
    <xsd:attribute name="timeStart" type="xsd:dateTime"></xsd:attribute>
    <xsd:attribute name="timeEnd" type="xsd:dateTime"></xsd:attribute>
  </xsd:complexType>
```

```
  <xsd:complexType name="timestamp">
    <xsd:attribute name="timestamp" type="xsd:dateTime" use="required">
    </xsd:attribute>
  </xsd:complexType>
</xsd:schema>
```

6 pXML Element Reference

This chapter is reference for the pXML schema elements. Some elements are in common to both pXML and DocXML and they appear in Chapter 11, *Common Elements*.

acl

Specifies an access control list (ACL). In general, an ACL consists of one or more group membership and permission pairs. For example, you could give the Administrators group discovery, read, write and ownership (DRWO) permissions. To do this, define one or more `aclSpecification` elements within your `acl` element.

Alternatively, you can reference an existing Palantir ACL. Typically, you would do this with the `aclReference`. However, you could use an `aclSpecification` and reconstruct the exact group membership and permissions. If you create an `aclSpecification` that matches an existing ACL, both reference the same ACL instance within the Workspace.

An `acl` can contain an `aclReference` or `aclSpecification` elements but not both.

Attributes

`aclId`	xsd:IDREF	R	Specifies the identifier used to reference this ACL elsewhere in a pXML document.

See Also

The `aclReference` element on page 52 and the `aclSpecification` element on page 53

Example

The following pXML example creates an ACL with the `Administrators` group having `discovery`, `read`, `write`, and `ownership` permissions and the `Everyone` group having `discovery` and `read` permissions. Note that the `authSourceTag` refers to internal Palantir groups.

```
<aclSet>
  <acl aclId="PT_ACL1">
    <aclSpecification authSourceTag="Palantir Internal Authentication"
        externalId="Administrators"
        type="group"
        permissions="owner"/>
  <aclSpecification authSourceTag="Palantir Internal Authentication"
        externalId="Everyone"
        type="group"
        permissions="read"/>
  </acl>
</aclSet>
```

aclReference

Specifies that the `acl` references an internal Palantir ACL instance. This is numeric ID inside of Palantir, not a reference to another `acl` ID in the document.

Attributes

`aclId`	long	R	The ID of an existing Palantir ACL.

Example

Elements referring to this value must refer to its `aclId`. For example, to refer to the ACL below an element would refer to `PT_ACL1`:

```
<aclSet>
  <acl aclId="PT_ACL1">
    <aclReference aclId=”PT_ACL3”/>
  </acl>
</aclSet>
```

aclSet

Lists the `acl` elements that the pXML references. You do not have to specify an `aclSet` in your pXML. If you do include it, you can supply a single, optional `defaultAcl` child elements and one or more `acl` elements. the system requires you to specify the `defaultAcl` child before any other `acl` elements.

See Also

The `defaultAcl` element on page 60 and the `acl` element on page 51.

aclSpecification

Defines the group and permissions associated with a particular access control level. One or more of these specifications make up an `acl`. When specifying permissions you can specify one of the following values (in increasing order):

- `discovery`
- `read`
- `write`
- `owner`

A higher permission level implies the lower levels. For example, giving a user write permissions automatically gives them read and discovery as well.

You must specify an authentication source known to Palantir's dispatch server. Palantir's own internal authentication service or you can use an alternate source known to the dispatcher. The Palantir administrator configures the known authentication sources. You can use the `authSourceConfigurator.sh` command to list the known authentication sources.

If you do not specify the `authSourceTag`, the system will attempt to locate the `externalId` across all known authentication sources. The Data Importer expects to locate only one matching authentication source. If it encounters 0 or more than 1 source, it reports an error and exits.

Attributes

`authSourceTag`	`nonEmptyString`	O	Specifies the authentication source of the group. If you wish to use a Palantir native group, specify "`Palantir Internal Authentication`" for this attribute. You might, for example, specify "`ldap`" as an authentication source. This tag is case sensitive.
`externalId`	`nonEmptyString`	R	The group's external ID.
`permissions`	string	R	The permissions for the ACI. This can take a value of `discovery`, `read`, `write`, or `owner`.

Example

See the `acl` element on page 51 for an example.

customKeyword

Specifies a set of additional terms on a `property` element. The system indexes these keywords along with the property's `propertyValue`. The system ensures the `customKeyword` value is tokenized and searchable but the system hides the actual keywords from the user.

Use a custom keyword to restrict searches related to an entity. For example, if you have an entity named Jane Reynolds you may want to disable the search on the keyword `Jane`. Such a search would return too many documents because Jane is such a common name.

The `property` element's `keywordDisable` attribute must be `false` for the system to include the `customKeyword` values in a search. If you do not specify this attribute on the `property` element, `keywordDisable` defaults to false.

Example

The following example restricts keyword searches on the `Name` property disabling searches on the simple first name of `Jane`.

```
<property type='com.palantir.property.Name'
          keywordDisable="true">
    <customKeyword >
      Jane
    </customKeyword>
  <propertyValue>
    <propertyData>Organization</propertyData>
  </propertyValue>
</property>
```

In this next example, you want to permit the keyword searches using the person's email:

```
<property id='PT_PROPERTY9' type='com.palantir.property.Email'
          linkType='com.palantir.link.Simple'
          role='com.palantir.role.none' keywordDisabled='false' >
  <propertyValue>
    <propertyComponent type='EMAIL_USERNAME' >
        <propertyData>jane</propertyData>
    </propertyComponent>
    <propertyComponent type='EMAIL_DOMAIN' >
        <propertyData>gmail.com</propertyData>
    </propertyComponent>
  </propertyValue>
```

Email addresses are unique so the number of matches returned from a search on this value is manageable.

dataSource

Identifies a data source that is input into Palantir. A `dataSource` is applied to object components such as properties, media, notes, or links (relationships). A data source can be any of the following:

structured	A database or any tabular delimited file.
semi-structured	An individual email or an email server.
unstructured	A document which may or may not be in XML format

A `dataSource` element identifies the pedigree or lineage external data, such as which news feed a piece of information came from. The data source you create is a child of the pXML data source that is created by importing this pXML. Each `dataSourceRecord` references one `dataSource`.

When you define a data source, you specify the source `type` value. The `type` value determines which icon the system uses when displaying the component in the Workspace. The `type` value also supplies some information during programmatic access to the data source. The types you will find yourself using most frequently are the following:

- `com.palantir.datasource.filetext`
- `com.palantir.datasource.tablespace`
- `com.palantir.datasource.filedocument`.

The following lists the set of all possible `type` values:

Value	Description
`com.palantir.datasource.table`	A generic DB table.
`com.palantir.datasource.filedocument`	Generic file system document (unstructured documents).
`com.palantir.datasource.filetext`	Generic structured text file (csv, etc.).
`com.palantir.datasource.user`	Manually entered data.
`com.palantir.datasource.user`	User created data source.
`com.palantir.datasource.email`	Email.
`com.palantir.datasource.excel`	Excel data sheet.

If you specify an unstructured `type`, your `dataSource` element must contain a `primaryObject` child element if you want to see the document in the document viewer. The data source title appears in the Palantir user interface. For unstructured data sources, the system uses the title from the primary object.

The Data Importer does not attempt to perform data source resolution. So, if it encounters two data sources with the same name, the importer does not merge them into one.

Attributes

`id`	xsd:ID	R	An identifier to the data source. Data source records within the pXML document use the id to reference this source. This identifier must be unique within your pXML but is not persisted in Palantir.
`aclId`	xsd:IDREF	O	The ACL assigned to this data source. This sets the permissions on the data source itself and any DSR's (Data Source Record) referencing this data source will inherit these permissions. The value of this field must match one of the `aclId` fields in the `aclSet` of this document.
`type`	XSD:ANYuri	R	The data source type.

Example

The following example illustrates the specification of an Excel spreadsheet data source:

```
<dataSourceSet>
  <dataSource id='PT_DATASOURCE1'
              type='com.palantir.datasource.exceldata' >
    <name>contacts</name>
    <description>
      Sheet 'contacts' in workbook 'contacts.xls'
    </description>
  </dataSource>
</dataSourceSet>
```

See Also

The `name` element on page 65, the `description` element on page 61, and the `primaryObject` element on page 69.

dataSourceRecord

Associates an object component with a data source. This element persists a data's pedigree and lineage information. The `dataSource` and `importKey` are two key attributes on a `dataSourceRecord`.

The `dataSource` attribute must reference a data source contained within the `dataSourceSet` element. The `importKey` and `stringPositionLocator` are user-definable fields that you can use to supply more specific information about the data's location in the source system.

The `importKey` is a string value that contains any type of information you like. For example, the `importKey` could contain the primary key of the original database row the data came from.

Finally, you can define a `stringPositionLocator` element as a child element. For an unstructured data source, you should specify a 0-length import key and then specify a unique string position locator (SPL). The `stringPositionLocator` element details the offsets in the unstructured source that pinpoint a data's location.

Note: It is an error to provide both a `stringPositionLocator` and a non-zero-length `importKey`.

Attributes

`dataSource`	xsd:IDREF	R	Reference to a `dataSource` element that is a member of the `dataSourceSet` element.
`importKey`	string	R	A user-defined, additional information to associate back to the source data. This may be an empty string or a value up to 1024 characters.

recordLocator	long	O	A user-defined value to assist in locating the information in the source data system.
aclId	xsd:IDREF	O	An ACL to apply to the data source record. This value overrides the ACL specified on the dataSourceRecordSet element for this element only.

Example

```
<dataSourceRecord dataSource="PT_DATASOURCE18" importKey="">
    <stringPositionLocator startPosition="843"
                           endPosition="863"
                           sentenceNumber="28"
                           paragraphNumber="33"/>
  ...
</dataSourceRecord>
```

See Also

The stringPositionLocator element on page 76.

dataSourceRecordSet

Contains a set of one or more dataSourceRecord elements that belong to a parent object component (property, media, note, link). This element is optional you need not specify it at all. You can only specify one dataSourcerecordSet per object.

See Also

The dataSourceRecord element on page 58.

dataSourceSet

Contains one or more `dataSource` elements identifying data sources that may be referenced in the pXML document. This element is optional.

See Also

The `dataSource` element on page 56.

defaultAcl

Specifies the default access control list (ACL) that applies to all `object` components (properties, media, notes, links) and data sources in the pXML document. The format of the ACL is the same as the `acl` element. Alternatively, you can simply reference an existing `acl` element within your pXML.

This element is optional but if you specify it, it must be the first element in the `aclSet`.

Attributes

`aclId`	xsd:IDREF	R	References an ACL elsewhere in the pXML document.

Example

See the `acl` element on page 51 for an example.

See Also

The `aclReference` element on page 52 and the `aclSpecification` element on page 53.

description

Contains a string describing an object or a data source. On an `object`, this element can have a maximum length of 4000 characters. On a `dataSource`, this element is restricted to no more than 1000 characters.

See Also

The `point` element on page 111.

graph

Identifies the XML file as containing pXML to the Data Importer. This element is required in all pXML documents. A graph element can contain four, optional child elements:

`aclSet`	Defines a set of ACL to apply to the contents of the pXML. You can override this set at other points within the pXML graph.
`dataSourceSet`	Contains one or more `dataSource` elements specifying data sources that may be referenced in the pXML document.
`objectSet`	The set of objects on the graph.
`linkSet`	The set of relationships on the graph.

Each of these optional elements can occur at most one time.

See Also

The `aclSet` element on page 53, the `dataSourceSet` element on page 60, the `objectSet` element on page 68, and the `linkSet` element on page 63.

link

Defines an object-to-object relationship that exists within the exported data. A link is described by its endpoints the `parentRef` and the `childRef`. The Data Importer will discard a link if either endpoint is not found in the exported object set.

You must specify a link `type` attribute. There is a set of intrinsic link types in the Palantir base ontology. However, your ontology may contain more than this if you have customized the possible link types. Use the Dynamic Ontology Manager to see a list of the relationships in your installation.

The importer discards a link if one either the child or parent reference is not in the pXML's `objectSet`.

Attributes

`parentRef`	xsd:IDREF	R	The ID of the parent (or higher-order object) endpoint in the link.
`childRef`	xsd:IDREF	R	The ID of the child (or lower-order object) endpoint in the link.
`type`	xsd:anyURI	R	The link's type, this must be an existing link type in the Palantir ontology.
`role`	xsd:anyURI	R	The role associated with the link.
`id`	xsd:ID	R	The ID of the link itself. This must be unique.
`aclId`	xsd:IDREF	O	A reference to the set of access controls that applies to the link. This is optional.

Example

```
<link id="link1" parentRef="id2" childRef="id3"
      role="com.palantir.role.none" type="com.palantir.link.ColleagueOf">
  <dataSourceRecordSet>
    <dataSourceRecord dataSource="id1" recordLocator="7" importKey="key2"
                      aclId="ACL_1"/>
  </dataSourceRecordSet>
</link>
```

See Also

The `dataSourceRecordSet` element on page 59.

linkSet

Contains a set of `link` elements that apply between object instances. If you specify this element, it must contain at least one `link`.

See Also

The `link` element on page 62.

media

Defines attached media content for an object. You can also use this to define a document's contents or an entity's mugshot. The following table lists the possible media types:

Image	Document	Data	Video	Audio	Other
PNG	PDF	Excel	AVI	WAV	ZIP
JPG	Word	Access	WMV	MP3	I2 Analysts Notebook File
GIF	Text		QuickTime	Ogg	XML
BMP	HTML		RealMedia	AU	PowerPoint
	Rich Text		MPEF	Matroska	Viso
	Word 2007		Ogg		
			Matroska		

Between media content and an object you define a `type` attribute. This attribute controls how Palantir interprets the media data:

`com.palantir.link.Simple`	Attach the media without any special handling
`com.palantir.link.Mugshot`	The media is the object's mugshot. If you use this type, the media should be image data.
`com.palantir.link.raw`	On a document, the media is the text contents displayed for the object.
`com.palantir.link.src`	On a document, the media is the object's original source.

Palantir infers the MIME type for the format from the media type you specify.

Attributes

`linkType`	xsd:anyURI	R	Specifies how Palantir interprets the media data.
`mediaType`	xsd:string	R	The content of the media for example video or audio.

See Also

The `mediaTitle` element on page 65, the `mediaShortDescription` element on page 65, the `mediaSet` element on page 65, th e`mediaData` element on page 64, and the `dataSourceRecordSet` element on page 59.

mediaData

Specifies the contents of the media file as a Base64 encoded binary large object (BLOB). If the media type is a text format, then the encoding of the underlying text must be UTF-8.

mediaSet

Contains the `media` associated with the parent object. This element is optional. If you provide a media element, you must specify at least one child `media` element.

See Also

The `media` element on page 63.

mediaDescription

Describes the contents of the `media` element in 4000 characters or less. This element is optional.

mediaShortDescription

Specifies the short description of the `media` element in 50 characters or length. Currently, Palantir does not display the short description to the user. This element is optional.

mediaTitle

Specifies the `media` element's title as the system displays it to the user. The system displays the title for the media's caption in the browser. This element is optional.

name

Sets the name for the `dataSource` element. This is a required element. This can be up 80 characters long.

nonEmptyString

Defines a simple element type. This has a minimum length of one character. Empty space is preserved.

nonNegativeLong

Specifies a long integer that must be at least a 1.

negativeOneLong

Specifies a negative long integer that must be at least a -1.

note

Creates a `note` element on the parent object. This element can contain the following child elements:

`noteTitle`	A string of up to 80 characters specifying the title. The system displays this value to the user.
`noteDescription`	An unlimited text string describing the note.
`dataSourceRecordSet`	The data sources associated with the note.

Attributes

`linkType`	xsd:anyURI	O	Deprecated. Specifies the note's link type to the parent object. If not provided, this defaults to `com.palantir.link.Simple`.
`aclId`	xsd:IDREF	O	A reference to the set of access controls that applies to the link. This is optional.

See Also

The `noteTitle` element on page 67, the `noteData` element on page 67, and the `dataSourceRecordSet` element on page 59.

noteData

Specifies a `note` element's text contents. This element can contain Wiki markup.

noteDescription

Contains a description of the note's contents. This is an unlimited string length.

noteSet

Contains the `note` elements associated with an object. This element is optional but if provided, there must be at least one child note.

noteTitle

Specifies the title of a `note` element. The system displays this value to the user. This element's value can be up to 80 characters.

object

The `object` element specifies the properties, notes and media (object components) associated with a Palantir object.

dataSourceRecord	An optional list of dataSource elements. You can only specify one of these.
description	An optional string element of 4000 characters or less.
mediaSet	An optional list of media elements. You can only specify one of these.
noteSet	An optional list of note elements. You can only specify one of these.
propertySet	An optional list of property elements. You can only specify one of these.
title	An optional string element of 80 characters or less.

Attributes

groupFlag	xsd:Boolean	O	A boolean that, when true, indicates that the object is a group. By default, an object typically represents a single entity and this value is false.
id	xsd:ID	R	Provides the unique identifier of the object within the pXML document. Palantir does not persist these IDs; they are used solely within the context of the pXML document.
type	xsd:anyURI	R	URI specifying the Palantir object type. This can be any type that exists in the current ontology.

See Also

The propertySet element on page 74, the mediaSet element on page 65, and the noteSet element on page 67.

objectSet

Contains the object elements defined in the pXML document. This element is required.

palantir

Represents the root for the document and must be present on all pXML documents.

See Also

The `graph` element on page 61.

primaryObject

Specifies that a `dataSource` has an object that represents the data associated with that source. This is an optional element. If you specify an unstructured data source, your `dataSource` element must contain a `primaryObject` child element. You would use this element when the `dataSource` represents for example, a cable.

The data source `title` appears in the Palantir user interface. For unstructured data sources, the system uses the `title` from the primary object.

The primary object contains the actual file data as media but by using a `primaryObject` you can source object components to the `dataSource`.

Attributes

`objectRef`	xsd:IDREF	R	Reference an object associated with the data source. The object must be within the pXML document.

Example

See the `stringPositionLocator` element on page 76 element for an example.

See Also

The dataSource element on page 56.

property

Defines an object attribute. You can represent any sort of information as a property but typically the attributes you represent include information like name, address, phone number, badge number and so forth. A property element can contain the following sequence of child elements:

customKeyword	Defines additional terms to index.
timestamp	Specifies a point in time that indicates when the property is relevant. You can specify this or timeInterval but not both.
timeInterval	Specifies a time interval during which the property is relevant. You can specify this or timeStamp but not both.
description	Specifies a geographical point.
propertyValue	Defines the data contained in the property.
dataSourceRecordSet	A list of dataSource elements that link to this property.

The child elements are order dependent. You must specify these child elements in the order listed above. If you specify a subset of the child elements, you must maintain the same relative order.

Every property element requires a type attribute. Palantir recognizes two special property types the com.palantir.property.IntrinsicTitle and the com.palantir.property.TimeInterval.

In most cases, you should use the title element. The IntrinsicTitle is a composite property that defines the object title or label that the system displays to a user. If you use this property type, then you must set the property's linkType attribute to com.palantir.link.Title. You must also include a propertyValue element with components of TITLE and PRIORITY. This ensures the title displays correctly; otherwise the title appears as an object property and not its label. You can specify multiple IntrinsicTitle properties and Palantir will display the one with the highest priority.

You define a `type` attribute of `com.palantir.property.TimeInterval` when defining start and end time on an object. You can define a point in time or a period in time when the parent object is relevant. Typically, you specify time intervals on event and document objects. You must set the property's `linkType` attribute to `com.palantir.link.TimeInterval` for the value to display correctly. Finally, the `propertyValue` must also contain a `propertyTimeInterval` child element.

Each `property` element has an optional `role` attribute. Roles can be one of the following:

- `com.palantir.role.none`
- `com.palantir.role.from`
- `com.palantir.role.to`

Typically, you will uses roles for objects that are events. For example, suppose a phone call event had a phone number property with a role of `from` and another phone number property with a role of `to`, then you could think of the phone call as going *from* one phone number *to* another.

Attributes

`type`	xsd:anyURI	R	A URI from the current Palantir ontology. Use the Palantir Dynamic Ontology Manager to view possible properties in your ontology.
`linkType`	xsd:anyURI	O	The link the property has to the object. If you do not specify this value, it defaults to `com.palantir.link.Simple`.
`role`	xsd:anyURI	O	URI specifying the role the property has with respect to the object. If you do not specify this property, it defaults to `com.palantir.role.none`.
`id`	xsd:ID	O	Unique identifier for the property. If present, this ID must be unique for the entire document. Palantir does not maintain this ID in the repository. So, if you are not referencing this ID, you can safely leave it out.
`keywordDisabled`	xsd:Boolean	O	Controls indexing of the `customKeyword` child element. When false, the system indexes these keywords. When this is true, it does not. By default, this is false.
`aclId`	xsd:IDREF	O	References an `acl` element in the pXML.

Example

The following example illustrates the definition of an object's title. Notice the correspondence between the `property` type of `IntrinsicTitle` and the `linkType` of `Title`. This ensures the system displays this `propertyValue` as the object's label.

```
<property type='com.palantir.property.IntrinsicTitle
          linkType='com.palantir.link.Title' >
  <propertyValue>
    <propertyComponent type='TITLE' >
      <propertyData>Organization</propertyData>
    </propertyComponent>
    <propertyComponent type='PRIORITY' >
      <propertyData>1000000000</propertyData>
    </propertyComponent>
  </propertyValue>
</property>
```

See Also

The `customKeyword` element on page 54, th e`propertyValue` element on page 76, the `dataSourceRecordSet` element on page 59.

propertyComponent

Contains the `propertyData` for a `propertyValue` on a `property` element. A property is either a simple type (string, number, date) or a composite type. You can determine which category a property falls into by examining the property's **Base type** field in the Ontology Manager.

A composite property is composed of multiple `propertyComponent` elements. Specify each component of the composite property only once. However, you need not specify all components, you can specify a subset.

Attributes

`type`	xsd:anyURI	R	The URI of the property component. This is usually a single token such as `TITLE` or `FIRST_NAME`.

Example

The following XML constructs a `com.palantir.property.Phone` property, which is a composite property:

```
<property type="com.palantir.property.Phone">
  <propertyValue>
    <propertyComponent type="COUNTRY_CODE">
      <propertyData>1</propertyData>
    </propertyComponent>
    <propertyComponent type="AREA_CODE">
      <propertyData>925</propertyData>
    </propertyComponent>
    <propertyComponent type="PHONE_NUMBER">
      <propertyData>9504905</propertyData>
    </propertyComponent>
  </propertyValue>
</property>
```

See Also

The `propertyData` element on page 73.

propertyData

Contains data associated with a `propertyValue` element. A property is either a simple type (string, number, date) or a composite type. In the case of a simple property, you place a `propertyData` element directly within the `propertyValue` element. In the case of a composite property, `propertyData` appears within the respective `propertyComponent` elements.

Example

The following example illustrates a simple property:

```
<property type="com.palantir.property.Nationality">
  <propertyValue>
    <propertyData>GB</propertyData>
  </propertyValue>
```

```
  <dataSourceRecordSet>
    <dataSourceRecord dataSource="PT_DATASOURCE2" importKey="2"
recordLocator="0">
    </dataSourceRecord>
  </dataSourceRecordSet>
</property>
```

See the `propertyComponent` example for an illustration of a composite property.

propertyRawValue

Provides a property's value.The system imports the property's value using the property's assigned parser. You can see the parsers applied to a property using the Ontology Manager. You can use this element regardless of whether the property is simple or composite.

Note: You can use the `propertyRawValue` tag in place of the `propertyData` tag.

Example

The following example is a phone number:

```
<property type="com.palantir.property.Phone">
  <propertyValue>
    <propertyRawValue>925-950-4905</propertyRawValue>
  </propertyValue>
</property>
```

propertySet

Contains all of an `object` element's properties. `propertySet` is optional—if you do not define it, Palantir determines the object has no properties. If it is present, there must be at least one child `property` element.

See Also

The `property` element on page 70.

propertyTimeInterval

Defines an interval or point in time when an `object` is relevant. Typically, you would define this element for an event or document object. To indicate an object has a time relevancy, you define a `property` with a `com.palantir.property.TimeInterval` type. Then, you set the property's `linkType` attribute to `com.palantir.link.TimeInterval` for the value to display correctly.

Within the `propertyValue` child element, you define a `propertyTimeInterval` element. The `timeStart` and `timeEnd` attributes in this element must conform to the ISO 8601 date/time format. You must specify either both of these attributes or one of them.

Attributes

`timeStart`	xsd:dateTIme	O	The time that the interval begins.
`timeEnd`	xsd:dateTime	O	The time that the interval ends.

Example

The following example is an event with a time interval specified.

```
<property type='com.palantir.property.TimeInterval'
          linkType='com.palantir.link.TimeInterval'>
  <propertyValue>
    <propertyTimeInterval timeStart='2008-03-01T08:30:00.000-08:00'
                          timeEnd='2008-03-01T13:30:00.000-08:00' />
  </propertyValue>
</property>
```

propertyValue

Defines the value associated with a `property` element. When constructing the pXML for this element, you must have an understanding of whether the property is simple (string, number, date) or composite. You can determine which category a property falls into by examining the property's **Base type** field in the Ontology Manager. Depending on the property type, this element may contain any of the following child elements:

`propertyData`	Use this for simple string and number properties such as profession or age.
`propertyComponent`	Use this for composite properties such as phone numbers, addresses, and so forth. Each `propertyComponent` will contain one or more `propertyData` elements.
`propertyRawValue`	Use this for simple or composite properties. If you supply a raw value for a composite property, Palantir will use the parser defined in the ontology.
`propertyTimeInterval`	Use this for simple date properties.

Having no child elements is allowed and will created an empty property. Though, this is not a recommended practice.

See Also

The `propertyData` element on page 73, the `propertyComponent` element on page 72, the `propertyRawValue` element on page 74, the `propertyTimeInterval` element on page 75.

stringPositionLocator

Used with tagged unstructured data to locate the property data to a specific location within a document. Use this element to pinpoint the character offset and sentence/ paragraph information of the data source record.

For an unstructured data source, you should specify a 0-length import key and then specify a unique string position locator (SPL). It is an error to provide both a `stringPositionLocator` and a non-zero-length `importKey`.

Attributes

`startPosition`	xsd:nonNegativeInteger	O	Specifies the starting character offset of the tag.
`endPosition`	xsd:nonNegativeInteger	O	Specifies the ending character offset of the tag.
`sentenceNumber`	xsd:nonNegativeInteger	O	Specifies the sentence number of the tag. This is used to calculate co-occurrence information.
`paragraphNumber`	xsd:nonNegativeInteger	O	Specifies the paragraph number of the tag. This is used to calculate co-occurrence information.

Example

```
<property type='com.palantir.property.EventTitle'>
  <propertyValue>
    <propertyData>Data import</propertyData>
  </propertyValue>
  <dataSourceRecordSet>
    <dataSourceRecord dataSource='PT_DATASOURCE1'importKey=''
                      recordLocator='0' >
      <stringPositionLocator startPosition='116'
                             endPosition='127'
                             sentenceNumber='0'
                             paragraphNumber='4' />
    </dataSourceRecord>
  </dataSourceRecordSet>
</property>
```

Deprecated Elements and Attributes

The following elements are deprecated.

- `realmId`
- `SecurityAttributes`

- `type` attribute on the `aclSpecification` element
- `timeEnd` and `timeStart` attributes on `object` element
- `thumbnail`
- `thumbnailData`
- `mimeType` and `filename` on the `media` element

7 Working with the DocXML Schema Map

This chapter contains a complete listing of the `PalantirDocXMLSchemaMap.xsd` document. It also provides an example of a working DocXML Schema Map..

PalantirDocXMLSchemaMap.xsd File

The following is the DocXML Schema Map in its entirety:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--$Id:PalantirXMLImportSchema.xsd 31048 2007-07-23 01:19:40Z regs $ -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.palantirtech.com/pg/schema/docxmlmap/"
  xmlns:tns="http://www.palantirtech.com/pg/schema/docxmlmap/"
  elementFormDefault="qualified">

  <xsd:complexType name="mapping">
    <xsd:sequence>
      <xsd:element name="documentPropertyMapping"
                   type="tns:documentPropertyMapping" minOccurs="0">
      </xsd:element>
      <xsd:element name="mappingContext" type="tns:mappingContext"
                    maxOccurs="unbounded">
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="mappingContext">
    <xsd:sequence>
      <xsd:element name="typeMapping" type="tns:typeMapping"
                    maxOccurs="unbounded">
```

```
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="externalID" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="typeMapping">
    <xsd:attribute name="externalType" type="xsd:string" use="required"/>
    <xsd:attribute name="palantirObjectType" type="xsd:anyURI"/>
    <xsd:attribute name="palantirPropertyType" type="xsd:anyURI"/>
    <xsd:attribute name="palantirLinkType" type="xsd:anyURI"/>
   <xsd:attribute name="nullMapping" type="xsd:boolean" default="false"/>
  </xsd:complexType>

  <xsd:element name="mapping" type="tns:mapping">
   </xsd:element>

  <xsd:complexType name="documentPropertyMapping">
    <xsd:sequence>
      <xsd:element name="propertyMapping" type="tns:propertyMapping"
                   minOccurs="0" maxOccurs="unbounded"></xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="propertyMapping">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="palantirPropertyType" type="xsd:anyURI"
                   use="required"/>
  </xsd:complexType>
</xsd:schema>
```

Understanding a DocXML Schema Map

Each Palantir installation has a DocXML Schema Map to support importing document data from external sources. The Schema Map is an XML format. If your DocXML contains external property, object, or relationship (link) types, the schema maps these into Palantir types. This section explains the structure of the map.

For information on installing a Schema Map and running the Schema Map Editor, see the *Palantir Administrator's Guide*.

The Schema Map and DocXML Generation

The purpose of the DocXML Schema Map is to allow DocXML files to be independent of any particular ontology. In fact, you could import the same DocXML file into two Palantir deployments with vastly different ontologies. Or, you could import DocXML generated by two different generators into the same ontology.

Each DocXML generator can have its own object, property, and link types. One DocXML generator can call a police officer a “bobby” and another can simply call that object type an “officer.” You can map output from both these generators into the same ontology using the Schema Map.

Within the Schema Map the tags unique to a DocXML generator appear in the `contextMapping` block. This block maps each object type, property, and relationship from the generator into a Palantir type. The `externalId` attribute on the `contextMapping` element identifies the generator that produces these types.

When implementing a system that will export DocXML, you should keep these things in mind:

- A DocXML generator has complete freedom in determining what object, property, and link types to express—and what to name them.
- All DocXML files from the same generator must use an identical set of object, property, and link (relationship) types.
- Every DocXML output from a generator must specify the same `externalID` within its `objectSetMetaData` element.

Finally, when implementing your system, you should try to keep the total number of possible object, property, and link types somewhere between 1 and 200 total types.

Example of a Simple Schema Map

The following example illustrates the DocXML Schema Map in use:

```
<?xml version="1.0" encoding="utf-8"?>
<mapping xmlns="http://www.palantirtech.com/pg/schema/docxmlmap/">
  <documentPropertyMapping>
    <propertyMapping name="ppl count"
            palantirPropertyType="com.palantir.property.PersonCount"/>
    <propertyMapping name="location"
            palantirPropertyType="com.palantir.property.Location"/>
  </documentPropertyMapping>
  <mappingContext externalID="docTest">
    <typeMapping externalType="person"
                palantirObjectType="com.palantir.object.Person" />
    <typeMapping externalType="member"
```

```
                palantirLinkType="com.palantir.link.MemberOf" />
    <typeMapping externalType="name"
                palantirPropertyType="com.palantir.property.Name" />
    <typeMapping externalType="aka"
                palantirPropertyType="com.palantir.property.aka" />
    <typeMapping externalType="unmapped" nullMapping="true" />
  </mappingContext>
</mapping>
```

The `documentPropertyMapping` block maps external document property types to Palantir types. Document property mappings are in a separate section because they are specific to an organization.

There can, however, be several different `mappingContext` sections, one for each DocXML generator you use. For example, you may have one mapping for data coming from a database and another context for data coming from an API feed. A single `typeMapping` can have any combination of Palantir object, link, and property types specified.

Having a null mapping suppresses an error from showing up on the UI when an external type does not exist. The data importer does not create:

- Properties with a null property type mapping.
- Objects with only null-mapped properties
- Objects with a null object type mapping. The importer ignores all the object's properties.
- Links with a null link type.

Review the example mapping provded for the `unmapped` type:

```
<typeMapping externalType="unmapped" nullMapping="true" />
```

Based on this mapping, wherever the system encounters an object, relationship (link) or property with the `unmapped` type, the system will ignore it.

If a null mapping is done for an external type with a object, link, or property mapping, any mappings to Palantir types left unspecified become null mappings. You could map the `name` property type to `com.palantir.property.Name` but still make `name` a null mapping as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<mapping xmlns="http://www.palantirtech.com/pg/schema/docxmlmap/">
  ...
  <mappingContext externalID="docTest">
    <typeMapping externalType="name"
                palantirPropertyType="com.palantir.property.Name" />
    <typeMapping externalType="name" nullMapping="true" />
  </mappingContext>
</mapping>
```

The system would map the property `name` to `com.palantir.property.Name` but would use the `name` null mapping for object and link types. You can also use the alternative construction for this which is:

```
<?xml version="1.0" encoding="utf-8"?>
<mapping xmlns="http://www.palantirtech.com/pg/schema/docxmlmap/">
  ...
  <mappingContext externalID="docTest">
    <typeMapping externalType="name"
                 palantirPropertyType="com.palantir.property.Name"
                 nullMapping=”true”/>
  </mappingContext>
</mapping>
```

This construction has the same effect using one element as the previous which uses two elements.

When to Edit the Schema Map

To change your DocXML Schema Map, you must use the Text Resource Editor. The Text Resource Editor CLI allows an administrator to edit the default Schema Map for DocXML. If you add a new DocXML generator, you will need to add a new mapping context to the Schema Map so that the DocXML can be properly imported.

See the *Palantir Administrator’s Guide* for more information on using the Text Resource Editor.

8 DocXML Schema Map Element Reference

This chapter is reference for the elements found in the DocXML Schema Map.

documentPropertyMapping

Maps external document property types to Palantir types. This element contains zero or more `propertyMapping` elements. The `propertyMapping` elements map an external property type to a Palantir internal type. These mappings are specific to an organization.

mapping

Top level element in the Schema map. This element contains two child elements the `documentPropertyMapping` element and the `mappingContext` element. You must have at least one `mappingContext` element. The `documentPropertyMapping` element is optional.

mappingContext

Defines mappings for each source of DocXML you use. The entity extractor uses the `mappingContext` element when tagging documents on import. This element contains one or more `typeMapping` child elements.

Each DocXML generator has a unique ID. This ID is reference by each generated DocXML file through the `externalId` attribute on the `objectSetMetaData` element. Your `mappingContext` must have its own `externalID` attribute that also references this ID.

Attributes

`externalID`	xsd:string	R	Identifies the generator of the DocXML file. This value must map to the `externalId` attribute on the DocXML file's `objectSetMetaData` element.

propertyMapping

Maps an external document property to an internal Palantir type. This mapping maps the `name` attribute specified in the `property` element to a Palantir internal type. The property element is a child of the `documentMetaData` element.

Note: For mapping the types found in objects, text references, or relationships use the `typeMapping` element.

Attributes

`name`	xsd:string	R	The external named type in the DocXML.
`palantirPropertyType`	xsd:anyURI	O	A Palantir object type in your ontology.

Example

In this example, assume that you have a DocXML such as the following:

```
<?xml version="1.0" encoding="utf-8"?>
<palantir xmlns="http://www.palantirtech.com/pg/schema/docxml/">
  <document>
    <documentMetaData>
      <title>Georgia Tech Recruiting Report 2</title>
      ...
      <documentProperties>
        <property name="ppl count" value="83"/>
        <property name="location" value="USA"/>
      </documentProperties>
...
```

You will need to map both the `ppl count` and the `location` property in your DocXML Schema Map. The following shows how this will look:

```
<?xml version="1.0" encoding="utf-8"?>
<mapping xmlns="http://www.palantirtech.com/pg/schema/docxmlmap/">
  <documentPropertyMapping>
    <propertyMapping name="ppl count"
            palantirPropertyType="com.palantir.property.PersonCount" />
    <propertyMapping name="location"
            palantirPropertyType="com.palantir.property.Location" />
  </documentPropertyMapping>
  <mappingContext externalID="docTest">
  ...
```

typeMapping

Maps an external object, text reference, or relationship (link) type to a Palantir object, link, or property type. A single `typeMapping` has three attributes that allow you to specify how a named external type maps to a Palantir type. You can have any combination of Palantir object, link, and property types specified.

You can also include a `nullMapping` attribute that takes `true` or `false`. When `true`, this attribute suppresses an error from showing up in the interface when an external type does not exist. The data importer does not create:

- Text references with a null property type mapping.
- Objects with only null-mapped properties.

- Objects with a null object type mapping. The importer ignores all of the object's properties.
- Links with a null link type mapping.

Attributes

`externalType`	xsd:string	R	The external named type in the DocXML.
`palantirObjectType`	xsd:anyURI	O	A Palantir object type in your ontology.
`palantirPropertyType`	xsd:anyURI	O	A Palantir property type in your ontology.
`palantirLinkType`	xsd:anyURI	O	A Palantir link type in your ontology.
`nullMapping`	xsd:boolean	O	Specifies whether the system should ignore external types that are not mapped. The default is false.

Examples

The following example maps objects with the name `person` to `com.palantir.object.Person` type. Links or properties named `person` map to null, so the system ignores them:

```
<typeMapping externalType="person"
            palantirObjectType="com.palantir.object.Person"
            nullMapping="true"/>
```

This example, maps `member` type links to the `com.palantir.link.MemberOf` type. The system ignores object or properties with an external type of `member`:

```
<typeMapping externalType="member"
            palantirLinkType="com.palantir.link.MemberOf"
            nullMapping = "true"/>
```

This next example, maps `name` type links to the `com.palantir.property.Name` type. This example sets `nullMappping` to false causing the system to return an error if object or properties with an external type of `name` appear in the DocXML:

```
<typeMapping externalType="name"
            palantirPropertyType="com.palantir.link.Name"
            nullMapping = "false"/>
```

In the example below, causes the import to ignore any object, link, or property with a name of `unmapped`:

```
<typeMapping externalType="unmapped" nullMapping="true" />
```

9 PalantirDocXMLSchema.xsd File

This chapter contains a complete listing of the `PalantirDocXMLSchema.xsd` file used for defining documents for import. For documentation on the individual elements see Chapter 10, *DocXML Element Reference*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.palantirtech.com/pg/schema/docxml/"
      xmlns:tns="http://www.palantirtech.com/pg/schema/docxml/"
      elementFormDefault="qualified">

  <xsd:element name="palantir" type="tns:palantir">
   </xsd:element>

  <xsd:complexType name="palantir">
    <xsd:sequence>
      <xsd:element name="document" type="tns:document"></xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="document">
    <xsd:sequence minOccurs="1">
      <xsd:element name="documentMetaData" type="tns:documentMetaData">
      </xsd:element>
      <xsd:element name="content" type="tns:content"></xsd:element>
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="objectData" type="tns:objectdata">
        </xsd:element>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="palantirType" type="xsd:anyURI"/>
  </xsd:complexType>

  <xsd:complexType name="documentMetaData">
    <xsd:sequence>
```

```
      <xsd:element name="title" type="xsd:string"></xsd:element>
      <xsd:element name="datasourceTitle" type="xsd:string"
            minOccurs="0">
      </xsd:element>
      <xsd:element name="source" type="xsd:anyURI"></xsd:element>
      <xsd:element name="contentType" type="xsd:string"></xsd:element>
      <xsd:element name="encoding" type="xsd:string"></xsd:element>
      <xsd:element name="timestamp" type="xsd:dateTime"></xsd:element>
      <xsd:element name="classification" type="xsd:string"
                  minOccurs="0">
      </xsd:element>
      <xsd:element name="documentProperties"
              type="tns:documentProperties" minOccurs="0"></xsd:element>
      <xsd:element name="sentenceBreaks" minOccurs="0">
        <xsd:complexType>
         <xsd:attribute name="offsets" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="paragraphBreaks" minOccurs="0">
        <xsd:complexType>
         <xsd:attribute name="offsets" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="acl" type="tns:acl" minOccurs="0"></xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="documentProperties">
    <xsd:sequence>
      <xsd:element name="property" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:choice minOccurs="0">
              <xsd:element name="timestamp" type="tns:timestamp" />
              <xsd:element name="timeInterval" type="tns:timeInterval" />
            </xsd:choice>
            <xsd:element name="gisData" type="tns:gisData"
                        minOccurs="0"></xsd:element>
          </xsd:sequence>
          <xsd:attribute name="name" type="xsd:string"/>
          <xsd:attribute name="palantirType" type="xsd:anyURI"/>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="acl">
    <xsd:sequence>
      <xsd:element name="aclSpecification" type="tns:aclSpecification"
                  maxOccurs="unbounded"></xsd:element>
    </xsd:sequence>
```

```
</xsd:complexType>

<xsd:complexType name="aclSpecification">
  <xsd:sequence/>
  <xsd:attribute name="authSourceTag" type="xsd:string"/>
  <xsd:attribute name="externalId" type="tns:nonEmptyString"
                 use="required">
  </xsd:attribute>
  <xsd:attribute name="permissions" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="discovery"/>
        <xsd:enumeration value="read"/>
        <xsd:enumeration value="write"/>
        <xsd:enumeration value="owner"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

  <xsd:simpleType name="nonEmptyString">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:whiteSpace value="preserve"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="content">
  <xsd:restriction base="xsd:base64Binary"/>
</xsd:simpleType>

<xsd:complexType name="objectSet">
  <xsd:sequence>
    <xsd:element name="objectSetMetaData" type="tns:objectSetMetaData">
    </xsd:element>
    <xsd:element name="objectSet" type="tns:objectSet"></xsd:element>
    <xsd:element name="relationshipSet" type="tns:relationshipSet"
                  minOccurs="0">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="objectSetMetaData">
  <xsd:sequence>
    <xsd:element name="description" type="xsd:string" minOccurs="0">
    </xsd:element>
    <xsd:element name="externalMetaData" type="xsd:base64Binary"
                 minOccurs="0">
    </xsd:element>
    <xsd:element name="original" type="xsd:base64Binary"
                  minOccurs="0">
    </xsd:element>
```

```
  </xsd:sequence>
  <xsd:attribute name="externalID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="objectSet">
  <xsd:sequence>
    <xsd:element name="object" type="tns:object" minOccurs="0"
                 maxOccurs="unbounded"></xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="object">
  <xsd:sequence>
    <xsd:element name="textReference" type="tns:textReference"
    maxOccurs="unbounded" minOccurs="0"></xsd:element>
  </xsd:sequence>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="palantirType" type="xsd:anyURI"/>
  <xsd:attribute name="objectID" type="xsd:ID"/>
  <xsd:attribute name="title" type="xsd:string"/>
  <xsd:attribute name="confidence" type="xsd:double"/>
</xsd:complexType>

<xsd:complexType name="textReference">
  <xsd:sequence>
    <xsd:element name="value" type="xsd:string"></xsd:element>
    <xsd:choice minOccurs="0">
      <xsd:element name="timestamp" type="tns:timestamp" />
      <xsd:element name="timeInterval" type="tns:timeInterval" />
    </xsd:choice>
    <xsd:element name="gisData" type="tns:gisData" minOccurs="0">
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="characterOffset" type="xsd:unsignedLong"
                 use="required">
  </xsd:attribute>
  <xsd:attribute name="characterLength" type="xsd:unsignedLong"
                  use="required">
  </xsd:attribute>
  <xsd:attribute name="propertyType" type="xsd:string"/>
  <xsd:attribute name="palantirPropertyType" type="xsd:anyURI"/>
  <xsd:attribute name="confidence" type="xsd:double"/>
</xsd:complexType>

<xsd:complexType name="timestamp">
  <xsd:attribute name="timestamp" type="xsd:dateTime" use="required">
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="timeInterval">
  <xsd:attribute name="timeStart" type="xsd:dateTime"></xsd:attribute>
  <xsd:attribute name="timeEnd" type="xsd:dateTime"></xsd:attribute>
```

```
  </xsd:complexType>

  <xsd:complexType name="gisData">
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:element name="point" type="tns:point"></xsd:element>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="point">
    <xsd:sequence minOccurs="0">
      <xsd:element name="userLatLong" type="tns:userLatLong"
                   minOccurs="0">
      </xsd:element>
      <xsd:element name="userUTM" type="tns:userUTM" minOccurs="0">
      </xsd:element>
      <xsd:element name="userMGRS" type="tns:userMGRS" minOccurs="0">
      </xsd:element>
      <xsd:element name="userElevation" type="tns:userElevation"
                   minOccurs="0">
    </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="latitude" type="xsd:double" use="required">
    </xsd:attribute>
    <xsd:attribute name="longitude" type="xsd:double" use="required">
    </xsd:attribute>
    <xsd:attribute name="elevation" type="xsd:double"></xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userLatLong">
    <xsd:attribute name="latitude" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="longitude" type="xsd:string" use="required">
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userUTM">
    <xsd:attribute name="utmZone" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="utmNorthing" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="utmEasting" type="xsd:string" use="required">
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userMGRS">
    <xsd:attribute name="mgrs" type="xsd:string" use="required"></
xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="userElevation">
```

```
      <xsd:attribute name="elevation" type="xsd:string" use="required">
      </xsd:attribute>
    </xsd:complexType>

      <xsd:complexType name="relationshipSet">
      <xsd:sequence>
        <xsd:element name="relationship" type="tns:relationship"
                     minOccurs="0" maxOccurs="unbounded">
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="relationship">
      <xsd:attribute name="childRef" type="xsd:IDREF" use="required"/>
      <xsd:attribute name="parentRef" type="xsd:IDREF" use="required"/>
      <xsd:attribute name="relationType" type="xsd:string"/>
      <xsd:attribute name="palantirRelationType" type="xsd:anyURI"/>
    </xsd:complexType>

  </xsd:schema>
```

10 DocXML Element Reference

This chapter is reference for the DocXML schema elements. Some elements are in common to both pXML and DocXML and they appear in Chapter 11, *Common Elements*.

acl

Specifies an access control list (ACL). You can define one or more `aclSpecification` elements within your `acl` element. This `acl` applies to the entire document its entities and relationships.

In general, an ACL consists of one or more group membership and permission pairs. For example, you could give the Administrators group owner permissions and Analysts write permissions. If you create an `aclSpecification` that matches an existing ACL, both reference the same ACL instance.

The `acl` must contain a permission set the gives the current user write permissions.

See Also

The `aclSpecification` element on page 96.

aclSpecification

Defines the group and permissions associated with a particular access control level. One or more of these specifications make up an `acl`. When constructing permissions you can specify one of the following values (in increasing order):

- `discovery`
- `read`
- `write`
- `owner`

The higher permission level implies the lower levels. For example, giving a user write permissions automatically gives them read and discovery as well.

You can specify a group-permissions pair from Palantir's authentication configuration or you can specify another authentication source. If you choose to use an another source, you must choose a source that is known to the dispatch server. The Palantir administrator configures the known authentication sources.

Attributes

`authSourceTag`	`nonEmptyString`	O	Specifies the authentication source of the group. If you wish to use a Palantir native group, specify "`Palantir Internal Authentication`" for this attribute. You might, for example, specify "`ldap`" as an authentication source. If you do not specify this value, the system searches all known authorization sources for a group with the specified `externalId` attribute. This tag is case sensitive.
`externalId`	`nonEmptyString`	R	Identifies the group's external ID. For example, if you are using LDAP this might be the domain name or the `objectGUID`.
`permissions`	string	R	Specifies the permissions for the ACI. This can take a value of `discovery`, `read`, `write`, or `owner`.

classification

Defines the Palantir classification level for this document. This is an optional string containing the level. You can use the Ontology Manager to view a list of the classification levels in your ontology.

content

Contains the document's content. This is an optional child element to the `document` tag. The `content` elements must contain a Base64 binary value.

See Also

The `document` element on page 99.

contentType

Identifies what kind of content makes up the document. This is an optional child on the `documentMetaData` element. The `contentType` value is a string. You should use the subtype from the encoding description. For example, this may be a value like `text/plain`.

You provide a string value for this element.

Example

```
...
<document>
  <documentMetaData>
    <title>Palantir Technologies</title>
    <source>PTEntityExtractionTutorial</source>
    <contentType>text/plain</contentType>
    <encoding>UTF-8</encoding>
    <timestamp>2008-04-15T23:10:15Z</timestamp>
```

```
  </documentMetaData>
...
```

See Also

The `documentMetaData` element on page 99.

datasourceTitle

Provides a title for the data source. This is an optional Palantir-specific extension. You can use this element to override titles on data source objects within Palantir. If you use this element it must follow the `title` in the `documentMetaData` element.

You provide a string value for this element.

See Also

The `documentMetaData` element on page 99 and the `title` element on page 109.

description

Describes the object set. This is an optional element. It is the first element in a sequence that appears as the first construction in the `objectSetMetaData` element.

document

Defines the document represented by the DocXML file. This element contains the following child elements:

`documentMetaData`	Data describing the document.
`content`	The documents content.
`objectData`	Describes object data that exists in the document.

Attributes

`palantirType`	xsd:anyURI	O	A URI for a Palantir document subtype. If you do not specify a valid Palantir document subtype or you omit this value, the system uses the intrinsic document type.

documentMetaData

Describes information about the document defined by the DocXML file. This tag contains the following child elements:

`title`	A string used as the title for the document and the data source. If this element exists, it must be the first in the `documentMetaData`.
`datasourceTitle`	A string containing an optional, data source title. This is Palantir-specific extension that overrides the title for the data source only.
`source`	A URI for the `com.palantir.property.Filename` document property. This is required.
`contentType`	A string denoting what kind of content the document contains.
`encoding`	A string identifying the code used on the document.
`timestamp`	A time interval on the document object. The Palantir Workspace Timeline uses this value.
`classification`	A string classifying the document.

`documentProperties`	An optional set of document properties.
`sentenceBreaks`	A list of offsets in the document detailing where the sentences break.
`paragraphBreaks`	A list of the offsets that correspond to paragraph breaks in the document.
`acl`	Specifies the access to the document.

The elements listed above make up an ordered sequence. If they all appear within `documentMetaData`, the must appear in the order show above. If a subset appears, they must maintain the relative order. So for example, if you supply a `title`, `paragraphBreaks`, and an `acl` element, the `paragraphBreaks` element must come after the `title` and before the `acl` element.

documentProperties

A list of the document `property` elements. This is an optional tag in a sequence of elements that appear within the `documentMetaData` tag.

encoding

Specifies how the document is encoded. You can specify any character set supported by the Java platform. Every implementation of the Java platform must support the following standard character sets:

US-ASCII	Seven-bit ASCII, also known as ISO646-US or the Basic Latin block of the Unicode character set.
ISO-8859-1	ISO Latin Alphabet No. 1, also known as ISO-LATIN-1.
UTF-8	Eight-bit UCS Transformation Format.
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order.
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order.
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark.

externalMetaData

Contains information about the document from an external source. This optional element contains Base64 binary content. The `externalMetaData` element appears within the `objectSetMetaData` element.

object

Details an entity related to or referenced within the document. The `object` element can contain one or more `textReference` elements. These elements contain the offsets for the reference. The `textReference` elements must appear first in a `object` definition.

Both the `palantirType` and the `type` attribute are optional. However, you must specify one or the other. If you specify both, the system uses the `palantirType` if it is valid. If it is not valid, the system uses the `type` attribute. If you specify a `type` attribute, you must ensure it is mapped in your system's DocXML Schema Map.

Attributes

`type`	xsd:string	O	A string specifying the object type. If you specify this value and a `palantirType` value, the `palantirType` value is used.
`palantirType`	xsd:anyURI	O	A valid URI for an existing Palantir type.
`objectID`	xsd:ID	O	A valid Palantir identifier for an object. If you specify an invalid or non-existent identifier, the importer will still import the document.
`title`	xsd:string	O	The object title.

objectData

Contains information about objects related to this document. This element has child elements that follow a well-defined sequence. The child elements have the following sequence:

`objectSetMetaData`	Contains information about the object data associated with the document.
`objectSet`	A list of related objects.
`relationshipSet`	A list of relevant relationships.

objectSet

Lists the objects associated with the document.

objectSetMetaData

Describes the object set associated with the document. This element contains the following child elements:

`description`	Describes the objects in the set. This is a string.
`externalMetaData`	Contains information about the objects from an external source. This is a Base64 binary.
`original`	Contains the original file in Base64 binary. For example, if the original file was a PDF file or Word document you could put its encoded contents here.

Attributes

`externalID`	xsd:string	O	An identifier for a DocXML generator. This `externalId` is reference from the `mappingContext` in the DocXML schema.

original

Contains the original document in Base64 binary. For example, if the original file was a PDF file or Word document you could put its encoded contents in the `original` element.

palantir

Represents the root of the DocXML document. There is one instance of this required tag and it contains a single `document` element.

See Also

The `document` element on page 99.

paragraphBreaks

Defines optional paragraph breaks in the `documentMetaData` element. A break is defined as the character offset (0-based) where paragraphs start. If you are tagging a entity that starts with the first letter in the document, set the `offsets` attribute to 0 (zero).

You can define sentence breaks and/or paragraph breaks. Using the paragraph and sentence breaks, the system merges the paragraph breaks into the list of sentence breaks so that one sentence does not span two paragraphs.

If you define paragraph breaks but not sentence breaks (or sentence and not paragraph breaks), the system generates default breaks for the missing values.

If you define paragraph breaks but not sentence breaks (or sentence and not paragraph breaks), the system generates default breaks for the missing values. In general, most DocXML does not specify paragraph breaks; the Data Importer applies breaks for you. You should permit this unless your export pipeline that uses them.

Attributes

`offsets`	xsd:string	R	A string containing the offsets. This is a comma-delimited string.

Example

```
<documentMetaData>
  <title>Georgia Tech Recruiting Report 2</title>
  <datasourceTitle>Different Datasource Title</datasourceTitle>
  <source>http://devblog.yojoe.local/</source>
  <contentType>text/plain</contentType>
  <encoding>UTF-8</encoding>
  <timestamp>1976-08-30T18:00:00Z</timestamp>
  <classification>UC</classification>
  <documentProperties>
    <property name="ppl count" value="83"/>
    <property palantirType="com.palantir.property.Location" value="USA"/>
  </documentProperties>
  <sentenceBreaks offsets="0, 2, 103, 107, 153, 1160">
  <paragraphBreaks offsets="0, 103, 107, 153, 174, 1160">
</documentMetaData>
```

See Also

The `documentMetaData` element on page 99 and the `sentenceBreaks` element on page 106.

property

Defines a document attribute. This element has the following child elements:

`timestamp`	An optional point in time that indicates when the `property` is relevant. You can specify this or `timeInterval` but not both.

`timeInterval`	An optional time interval during which the `property` is relevant. You can specify this or `timeStamp` but not both.
`gisData`	An optional geographical point. Use this to associate geographical data with a text reference.

You need not specify all the child elements. However, the elements are order dependent, for example, you cannot specify `gisData` before the `value` attribute.

You must supply either a `name` attribute or a `palantirType` attribute or both to the importer. If you supply both, the system uses the `palantirType`. If you specify the `name` attribute you must supply a DocXML schema that maps the `name` attribute into a Palantir property type.

Unlike pXML, DocXML does not have a special format for composite or simple properties. The system parses the `value` using the parsers associated with the Palantir type defined for the value.

Attributes

`name`	xsd:string	O	A string containing a property name.
`palantirType`	xsd:anyURI	O	A pointer to an existing Palantir property type.
`value`	xsd:string	R	The property's value.

relationship

Specifies a member of a `relationshipSet` element. You supply a `childRef` and a `parentRef` for each side of the relationship. The identifiers you supply must reference objects within the DocXML.

You must supply either a `relationType` attribute or a `palantirRelationType` attribute or both to the importer. If you supply both, the system uses the `palantirRelationType`. If you supply a `relationType` attribute you must ensure the DocXML schema maps the `relationType` attribute into a Palantir property type.

Attributes

childRef	xsd:IDREF	R	A reference to the child object.
parentRef	xsd:IDREF	R	A reference to the parent object.
relationType	xsd:string	O	A relationship name. This value must be mapped in your installation's DocXML Schema Map.
palantirRelationType	xsd:anyURI	O	A Palantir relationship. This relationship must already exist in your ontology.

relationshipSet

Contains a list of relationships represented within the document's data.

sentenceBreaks

Defines optional paragraph breaks in the documentMetaData element. A break is defined as the character offset (0-based) where paragraphs start. If you are tagging a entity that starts with the first letter in the document, set the offsets attribute to 0 (zero).

You can define sentence breaks and/or paragraph breaks. Using the paragraph and sentence breaks, the system merges the paragraph breaks into the list of sentence breaks so that one sentence does not span two paragraphs.

If you define paragraph breaks but not sentence breaks (or sentence and not paragraph breaks), the system generates default breaks for the missing values. In general, most DocXML does not specify sentence breaks; the Data Importer applies breaks for you. You should permit this unless your export pipeline that uses them.

Attributes

offsets	xsd:string	R	A string containing the offsets.

source

Specifies the location of the data source. This is any valid URI. The source element is required.

Example

The source is a file on a local disk:

```
...
<documentMetaData>
  <title>Cable123: CT Blue on AL-MUJHA Activity in Maddi</title>
  <datasourceTitle>Cable Traffic on AL-MUJHA Activity in Maddi</
datasourceTitle>
  <source>file:/C:/foobar.xml</source>
  <contentType>text/plain</contentType>
  ...
```

The source is an internet document:

```
...
<documentMetaData>
  <title>Georgia Tech Recruiting Report 2</title>
  <datasourceTitle>Different Datasource Title</datasourceTitle>
  <source>http://devblog.yojoe.local/</source>
  <contentType>text/plain</contentType>
  <encoding>UTF-8</encoding>
...
```

textReference

Details where an `object` is mentioned within a document and defines the associated property represented by the mention. The `textReference` element is optional. There is no limit on number of text references. This element has the following child elements:

value	The content of the reference. This element is required.
timestamp	An optional point in time that indicates when the `property` is relevant. You can specify this or `timeInterval` but not both.

timeInterval	An optional time interval during which the `property` is relevant. You can specify this or `timeStamp` but not both.
gisData	An optional geographical point. Use this to associate geographical data with a text reference.

You need not specify all the child elements. However, the elements are order dependent, for example, you cannot specify `gisData` before the `value` attribute.

You must supply either a `propertyType` attribute or a `palantirPropertyType` attribute or both to the importer. If you supply both, the system uses the `palantirPropertyType`. The `propertyType` attribute is intended for batch imports that you run from the command line. You must ensure the DocXML schema maps the `propertyType` attribute into a Palantir property type.

Attributes

`characteroffSet`	xsd:unsignedLong	R	Zero-indexed character (not byte) offset into the content of this document.
`characterLength`	xsd:unsignedLong	R	Length in characters of this reference.
propertyType	xsd:string	O	A relationship name. This value must be mapped in a schema document you supply when importing.
palantirPropertyType	xsd:anyURI	O	A Palantir relationship. This value must already exist in your ontology.

Example

```
<objectSet>objectSetMetaData
  <object type="Meeting" objectID="OBJECT1">
    <textReference characterOffset="365" characterLength="13"
propertyType="EventName">
      <value>Charity Event</value>
    </textReference>
    <textReference characterOffset="382" characterLength="5"
propertyType="Location">
      <value>Maddi</value>
    </textReference>
  </object>
  ....
```

```
</objectSet>
```

title

Specifies the title to apply to the document as a string. The value of the `title` element shows up in the Workspace as the document's label. This is a child of the `objectSetMetaData` element.

11 Common Elements

This chapter is a reference of all the complex and simple elements that are structurally the same in both pXML and DocXML.

nonEmptyString

Defines a simple element type. This has a minimum length of one character. Empty space is preserved.

gisData

Contains geographic data that is associated with a `property`. This is an optional element. There can be only one for each `property` definition.

See Also

`point` element on page 111

point

Represents a GIS coordinate. The `point` element is contained within a `gisData` element. The `point` element has the following optional child elements:

`userElevation`	Specifies the elevation of the specified point.
`userLatLong`	Defines latitudinal and longitudinal data making up a point.
`userMGRS`	Specifies the Military Grid Reference System (MGRS) value for a given point.
`userUTM`	Specifies the Universal Transverse Mercator (UTM) value for a given point.

If you specify all or some of these values, you must supply them in the order listed above. If, for example, you only have a `userMGRS` value for the point, you can supply that.

Alternatively, you can specify a `longitude`, `latitude`, or `elevation` attribute on the `point` itself and none of the child elements. Use whichever values are appropriate for your source data.

Attributes

`latitude`	xsd:double	O	An angular distance North or south of the equator.
`longitude`	xsd:double	O	An angular distance East and West of a line drawn between the North and South Poles and passing through Greenwich, England
`elevation`	xsd:double	O	The distance above the longitude and latitude.

Example

```
<point>
  <userMRGS mgrs="10SZ135325"/>
</point>
```

See Also

The `gisData` element on page 110.

timeInterval

Defines an interval in time applied to a specific property. Typically, you would define time intervals for properties on an event or document object. `timeInterval` can be closed or open. Closed is when you supply both the `timeStart` and `timeEnd` attributes. When you specify only one value, that is an open `timeInterval` meaning until this date (`timeEnd`) or since this date (`timeStart`).

The `timeStart` and `timeEnd` attributes must conform to the ISO 8601 date/time format. You must specify either both of these attributes or one of them.

Attributes

`timeStart`	xsd:dateTIme	O	The time at which the interval begins.
`timeEnd`	xsd:dateTime	O	The time at which the interval ends.

timestamp

Defines an point in time applied to an `object` or `property`. Typically, you would define time intervals for properties on an event or document object. The `timeStamp` element must conform to the ISO 8601 date/time format.

userElevation

Indicates the elevation at which a geographical `point` resides. You can use this value instead of the `elevation` attribute on the `point` element.

Attributes

elevation	xsd:string	R	The elevation value.

userLatLong

Specifies the latitude and longitude for a given geographical point. You can use this value instead of the latitude and longitude attributes on the point element.

Attributes

latitude	xsd:string	R	An angular distance North or south of the equator.
longitude	xsd:string	R	An angular distance East and West of a line drawn between the North and South Poles and passing through Greenwich, England

userMGRS

Specifies a Military Grid Reference System (MGRS) value for a point element. You can use this value instead of the latitude and longitude attributes on the point element.

Attributes

mgrs	xsd:string	R	The MGRS value.

userUTM

Specifies a Universal Transverse Mercator (UTM) value for a `point` element. You can use this value instead of the `latitude` and `longitude` attributes on the `point` element.

Attributes

`utmZONE`	xsd:string	R	The zone number value.
utmNorthing	xsd:string	R	The projected distance of the point from the equator.
utmEasting	xsd:string	R	The projected distance of the point from the central meridian.

Index