



Relevance Schemes Guide

Version 1.1

Symplectic

August 20, 2019

Contents

1	Introduction	3
1.1	Components of a relevance scheme file	5
2	The <code><rel:category></code> element	7
2.1	Introduction	7
2.2	Contents	7
3	The <code><rel:masks></code> element	8
3.1	Introduction	8
3.2	<code><rel:mask></code> element	8
3.3	Masking objects by type	9
3.3.1	<code><rel:exclude-objects-of-types></code>	9
3.3.2	<code><rel:exclude-objects-of-types-except></code>	9
3.3.3	<code><rel:inspect-object></code>	10
3.4	Masking objects by source	10
3.4.1	<code><rel:exclude-objects-with-source></code>	10
3.4.2	<code><rel:exclude-objects-without-source></code>	11
3.4.3	<code><rel:inspect-object></code>	11
3.5	Masking sources	11
3.5.1	<code><rel:exclude-sources></code>	12
3.5.2	<code><rel:exclude-sources-except></code>	12
3.5.3	<code><rel:inspect-each-object-record></code>	12
3.6	Masking fields	13
3.6.1	<code><rel:exclude-fields></code>	13
3.6.2	<code><rel:exclude-fields-except></code>	14
3.6.3	<code><rel:exclude-source-fields></code>	14
3.6.4	<code><rel:inspect-each-object-record></code>	14
3.7	Inspecting objects	15
3.7.1	<code><rel:inspect-object></code>	15
3.8	Inspecting object records	16
3.8.1	<code><rel:inspect-each-object-record></code>	16
4	The <code><rel:aggregations></code> element	18
4.1	Introduction	18
4.2	<code><rel:aggregation></code> element	18
5	The <code><rel:post-aggregation-masks></code> element	20
5.1	Introduction	20
5.2	<code><rel:post-aggregation-mask></code> element	20
5.2.1	<code><rel:inspect-aggregated-object></code>	20
6	The <code><rel:hash-definitions></code> element	22
6.1	Introduction	22
6.2	<code><rel:hash-definition></code>	22

6.2.1	Hashing link data - <code><rel:link-types></code> element	23
7	The <code><rel:relevance-definition-selector></code> element	24
7.1	Introduction	24
7.2	<code><rel:relevance-definition-selector></code>	24
7.2.1	<code><rel:relevance-definition-selection></code>	25
A	Logic Expressions	27
A.1	Conditions - the <code><rel:condition></code> element	27
A.1.1	Condition data types	27
A.1.2	Condition operators - the operator attribute	28
A.1.3	Condition operator support by data type	29
A.1.4	Mapping Elements data types to Condition data types	30
A.2	Results - the <code><rel:result></code> element	32
A.3	The <code><rel:if></code> element	32
A.4	The <code><rel:choose></code> element	34
B	Property Names and Data-parts	36
B.1	Object Property Names	36
B.2	Record Property Names	36
B.3	Record Field Names	36
B.3.1	Publication field Names	37
B.4	data-part Names	39
B.5	User Property Names	40
C	Alternative File Structure (Obsolete)	42
C.1	Components of alternative file structure	42
D	Best Practices when writing relevance schemes	43
D.1	Design for maintenance	43
D.1.1	Keep things simple	43
D.1.2	Use meaningful names	44
D.1.3	Use comments	44
D.2	Design for performance	45
D.2.1	Use <code><masks></code> in preference to <code><post-aggregation-masks></code>	45
D.2.2	Avoid <code><inspect-each-object-record></code> if possible	45
D.2.3	Do not include excess fields in hash definitions	46

1 Introduction

This document describes how to define relevance schemes.

Relevance schemes are used when data is being automatically pushed from Elements to an external data source, for example when pushing metadata changes to a repository or researcher profile data to an academic profile system.

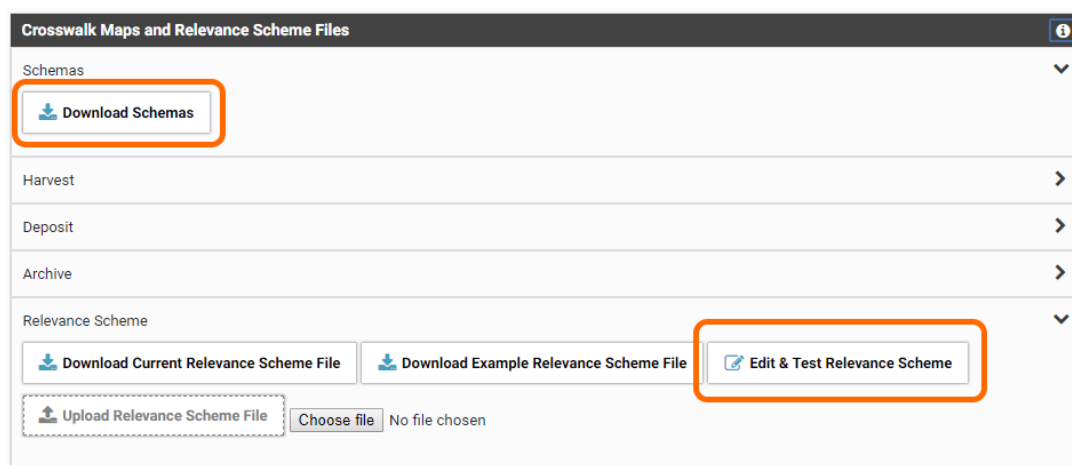
The purpose of a relevance scheme is to:

- identify whether an object is relevant to the external system (i.e. should the object be pushed to the external system or not)
- detect whether the data to be pushed to the external system has changed since the last push was made.

Each data source that supports automated pushing of data has its own relevance scheme. This is defined using a single XML file, that must conform to a standard XML schema. You can download this schema file (XmlRelevanceScheme.xsd) from Elements, by navigating to:

System Admin → Data → Data Source Management

and clicking the appropriate data source. Then scroll down to 'Crosswalk Maps and Relevance Scheme Files', expand the 'Schemas' section and click 'Download Schemas'.



Sample relevance scheme files are provided with Elements. **In order to avoid accidentally pushing data incorrectly these sample files are purposely designed to ensure that no data will be relevant for pushing.** Thus Elements administrators will need to edit these files to ensure they reflect the data structures and processes used in their organisation.

Elements provides a web page to enable authorised users to view, edit, test and update these XML files - expand the 'Relevance Scheme' section and click 'Edit & Test Relevance Scheme'.

This document describes how relevance schemes are defined at a detailed level. For an overview of Repository Tools 2 and data pushing, see the following articles on the Symplectic support

site:

- [Repository Tools 2: Configuring Automatic Metadata Updates including Relevance Schemes](#)
- [Repository Tools 2: Defining Relevance Schemes Guide.](#)

1.1 Components of a relevance scheme file

The structure of a relevance scheme file is shown below.

```
<xml-relevance-scheme
  xmlns="http://www.symplectic.co.uk/elements/xml-relevance-scheme">

  <!-- required element for each relevant category. -->
  <category name="publication">

    <masks>
      <!-- Optional element. Contains named masks that enable data that is never to
        ↳ be pushed to be masked. -->
    </masks>

    <aggregations>
      <!-- Optional element. Contains named aggregations which specify the source
        ↳ precedence to be used to identify a field value. -->
    </aggregations>

    <post-aggregation-masks>
      <!-- Optional element. Contains named masks that can perform further masking
        ↳ following aggregation. -->
    </post-aggregation-masks>

    <hash-definitions>
      <!-- Required element. Contains named hash definitions that are used to detect
        ↳ data changes. -->
    </hash-definitions>

    <relevance-definition-selector>
      <!-- Required element. Supports the selection of a named mask, aggregation,
        ↳ post-aggregation mask and hash-definition for an object. -->
    </relevance-definition-selector>

  </category>
</xml-relevance-scheme>
```

Note: An alternative format of relevance file is also supported, to maintain compatibility with some older implementations. Details of this format can be found in [Alternative File Structure \(Obsolete\)](#).

Each of the top-level elements is discussed in its own section of the documentation:

- [The <rel:category> element](#)
- [The <rel:masks> element](#)
- [The <rel:aggregations> element](#)
- [The <rel:post-aggregation-masks> element](#)

- The `<rel:hash-definitions>` element
- The `<rel:relevance-definition-selector>` element.

2 The `<rel:category>` element

2.1 Introduction

The `<rel:category>` element defines relevance and hash definitions for a specified category (using the `name` attribute).

```
<rel:category name="publication">
  ...
</rel:category>
```

An object can only be relevant if its category has a `<rel:category>` element present - all objects of categories which are not present are deemed to be not relevant. Thus it is important that `<rel:category>` elements are defined for all categories of object that could be relevant.

Note: The Elements infrastructure ensures that relevance schemes used for the automated push of RT2 repository metadata only receive publications which have an active repository record (not withdrawn or deleted).

2.2 Contents

The `<rel:category>` element contains XML elements that define how to assess relevance and calculate hashes for objects of the named category. These include:

- `<rel:masks>` element (see [Section 3](#))
Optionally defines masks that can be used to mask (hide) data so that it is excluded from further relevance assessment, hashing or pushing
- `<rel:aggregations>` element (see [Section 4](#))
Optionally defines the source precedence that is used when choosing which field value to use. If not present the default data source precedence is used.
- `<rel:post-aggregation-masks>` element (see [Section 5](#))
Optionally defines inspection rules to be applied following aggregation.
- `<rel:hash-definitions>` element (see [Section 6](#))
Defines hash definitions which determine whether a relevant change has been made.
- `<rel:relevance-definition-selector>` element (see [Section 7](#))
Inspects each supplied object and chooses which named mask, aggregation, post-aggregation mask and hash definition are to be used. If none are chosen, the object is deemed to be not relevant.

3 The `<rel:masks>` element

3.1 Introduction

Masks can be used to achieve the following two objectives:

- To recognise that an object is not relevant, by masking the entire object or all of its records
- To hide some data values associated with an object from being used in further relevance, hashing or pushing processes. This may be required for reasons of confidentiality, licencing or data quality.

The optional `<rel:masks>` element contains one or more uniquely named `<rel:mask>` elements. Each `<rel:mask>` element contains a set of rules that are applied to an object, in sequence, to mask (hide) some or all data values.

If the whole object is masked, or all records of the object are masked, the object is deemed to be not relevant.

The name of the mask to be applied to any particular object is defined using the `<rel:relevance-definition-selector>` (see [Section 7](#)).

A simple example of a `<rel:masks>` element is shown below:

```
<rel:masks>
  <rel:mask name="only-journal-articles-are-relevant">
    <rel:exclude-objects-of-types-except>
      <rel:type name="journal-article" />
    </rel:exclude-objects-of-types-except>
  </rel:mask>
</rel:masks>
```

This contains a single `<rel:mask>` element, named *only-journal-articles-are-relevant*, which excludes all objects unless they have a type of "journal-article".

3.2 `<rel:mask>` element

Each `<rel:mask>` element defines a mask that can be applied to an object when it is being assessed for relevance. A mask contains one or more rules that are applied to the object, in sequence. The rule may mask the whole object or part of the object (e.g. a record, a field or a field of a record).

The mask rules are viewed as either [Simple Mask Rules](#) or [Inspection Mask Rules](#).

Simple Mask Rules

Simple mask rules provide a mechanism to support the most common types of masking using a straightforward syntax. These rules support the following functions:

- Masking objects by type (see [Section 3.3](#))
- Masking objects by source (see [Section 3.4](#))
- Masking sources (see [Section 3.5](#))
- Masking fields (see [Section 3.6](#))

Inspection Mask Rules

Inspection rules provide a flexible mechanism to build logic statements that can be used to control how an object is masked. These rules support:

- Inspecting objects (see [Section 3.7](#))
- Inspecting object records (see [Section 3.8](#)).

Note that it may often be possible, and easier, to use multiple simple mask rules than to use inspection rules. However there are many cases where simple mask rules do not provide required functionality - consider for instance a requirement to mask all journal-articles which have no approved authorship links.

3.3 Masking objects by type

Objects can be masked based on their type. This can be achieved using the following elements:

- `<rel:exclude-objects-of-types>`
- `<rel:exclude-objects-of-types-except>`
- `<rel:inspect-object>`.

3.3.1 `<rel:exclude-objects-of-types>`

The `<rel:exclude-objects-of-types>` element provides the ability to specify the names of each object type that is to be excluded (i.e. objects of the specified types are not relevant).

This example will exclude all objects which have a type of *dataset* or *report*:

```
<rel:exclude-objects-of-types>
  <rel:type name="dataset"/>
  <rel:type name="report"/>
</rel:exclude-objects-of-types>
```

3.3.2 `<rel:exclude-objects-of-types-except>`

The `<rel:exclude-objects-of-types-except>` element provides the ability to specify the names of each object type that is not to be excluded (i.e. objects of the specified types may be relevant, objects of all other types are not relevant).

This example will exclude all objects except those which have a type of *journal-article* or *conference*:

```
<rel:exclude-objects-of-types-except>
  <rel:type name="journal-article"/>
  <rel:type name="conference"/>
</rel:exclude-objects-of-types-except>
```

3.3.3 <rel:inspect-object>

The <rel:inspect-object> element provides the ability to use [logic expressions](#) to determine how masking is to be applied to an object.

This example statement below achieves the same result as the example above (see [Section 3.3.2](#)).

```
<rel:inspect-object>
  <rel:if>
    <rel:condition operator="and">
      <rel:condition argument-field="object.type"
        ↪ operator="not-equals">journal-article</rel:condition>
      <rel:condition argument-field="object.type"
        ↪ operator="not-equals">conference</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:mask-object/>
    </rel:result>
  </rel:if>
</rel:inspect-object>
```

3.4 Masking objects by source

Objects can be masked based on whether they have data from a specific source. This can be achieved using:

- <rel:exclude-objects-with-source>
- <rel:exclude-objects-without-source>
- <rel:inspect-object>.

3.4.1 <rel:exclude-objects-with-source>

The <rel:exclude-objects-with-source> element provides the ability to exclude an object (i.e. deem it not relevant) if it contains any records from a list of named sources.

This example will exclude all objects which have an *arxiv* or a *dblp* record:

```
<rel:exclude-objects-with-source>
  <rel:source name="arxiv"/>
  <rel:source name="dblp"/>
</rel:exclude-objects-with-source>
```

3.4.2 <rel:exclude-objects-without-source>

The `<rel:exclude-objects-without-source>` element provides the ability to exclude an object (i.e. deem it not relevant) if it does not contain records from each source in a list of named sources.

This example will exclude all objects which do not have both a *scopus* and a *wos* record:

```
<rel:exclude-objects-without-source>
  <rel:source name="scopus"/>
  <rel:source name="wos"/>
</rel:exclude-objects-without-source>
```

3.4.3 <rel:inspect-object>

The `<rel:inspect-object>` element provides the ability to use [logic expressions](#) to determine how masking is to be applied to an object.

This example statement below achieves the same result as the example above (see [Section 3.4.2](#)).

```
<rel:inspect-object>
  <rel:if>
    <rel:condition operator="not">
      <rel:condition argument-field="object.record-sources"
        ↪ operator="contains-all-of">scopus,wos</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:mask-object/>
    </rel:result>
  </rel:if>
</rel:inspect-object>
```

3.5 Masking sources

Records from specified sources can be masked. This can be achieved using:

- `<rel:exclude-sources>`
- `<rel:exclude-sources-except>`
- `<rel:inspect-each-object-record>`.

Note that when relevance schemes are being used for RT2 metadata updates you should not mask the RT2 data source (but make sure it is not the top precedence data source - see [Aggregations](#)).

3.5.1 `<rel:exclude-sources>`

The `<rel:exclude-sources>` element provides the ability to exclude records from specified sources.

This example will exclude all records from *arxiv* and *repec*:

```
<rel:exclude-sources>
  <rel:source name="arxiv"/>
  <rel:source name="repec"/>
</rel:exclude-sources>
```

3.5.2 `<rel:exclude-sources-except>`

The `<rel:exclude-sources-except>` element provides the ability to exclude all records apart from those of specified sources.

This example will exclude all records apart from those from *dimensions*, *wos* and *local source 1 (c-inst-1)*:

```
<rel:exclude-sources-except>
  <rel:source name="dimensions"/>
  <rel:source name="wos"/>
  <rel:source name="c-inst-1"/>
</rel:exclude-sources-except>
```

3.5.3 `<rel:inspect-each-object-record>`

The `<rel:inspect-each-object-record>` element provides the ability to use [logic expressions](#) to determine how masking is to be applied to an object and its records.

This example statement below achieves the same result as the example above (see [Section 3.5.2](#)).

```

<rel:inspect-each-object-record>
  <rel:if>
    <rel:condition operator="and">
      <rel:condition argument-field="record.source"
        ↪ operator="not-equals">dimensions</rel:condition>
      <rel:condition argument-field="record.source"
        ↪ operator="not-equals">wos</rel:condition>
      <rel:condition argument-field="record.source"
        ↪ operator="not-equals">c-inst-1</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:mask-record/>
    </rel:result>
  </rel:if>
</rel:inspect-each-object-record>

```

3.6 Masking fields

Fields represent the individual values that are stored in each record. For example, the *abstract* field may have separate values defined (or not present) in the dimensions, wos and scopus records.

It is possible to mask all values of a field (e.g. if your institution does not want the values to be shown in the destination system) or the value that is stored in records from selected sources (e.g. if your institution is not licensed to use the data from a given source in the destination system, or does not trust the quality of the data from a specific source).

This can be achieved using:

- `<rel:exclude-fields>`
- `<rel:exclude-fields-except>`
- `<rel:exclude-source-fields>`
- `<rel:inspect-each-object-record>`.

3.6.1 `<rel:exclude-fields>`

The `<rel:exclude-fields>` element provides the ability to exclude all values of one or more fields, irrespective of source.

This example will exclude all *author-url* fields:

```

<rel:exclude-fields>
  <rel:field name="author-url"/>
</rel:exclude-fields>

```

3.6.2 <rel:exclude-fields-except>

The <rel:exclude-fields-except> element provides the ability to exclude all values of fields, irrespective of source, apart from specified fields.

This example will exclude all fields, apart from *title*, *abstract*, *authors*, *doi* and *publication-date*:

```
<rel:exclude-fields-except>
  <rel:field name="title"/>
  <rel:field name="abstract"/>
  <rel:field name="authors"/>
  <rel:field name="doi"/>
  <rel:field name="publication-date"/>
</rel:exclude-fields-except>
```

3.6.3 <rel:exclude-source-fields>

The <rel:exclude-source-fields> element provides the ability to mask a field of a specified source.

This example will exclude *abstract* from the *scopus* source:

```
<rel:exclude-source-fields>
  <rel:source-field source-name="scopus" field-name="abstract"/>
</rel:exclude-source-fields>
```

3.6.4 <rel:inspect-each-object-record>

The <rel:inspect-each-object-record> element provides the ability to use [logic expressions](#) to determine how masking is to be applied to an object and its records.

This example statement below achieves the same result as the example above (see [Section 3.6.3](#)).

```
<rel:inspect-each-object-record>
  <rel:if>
    <rel:condition argument-field="record.source"
      ↪ operator="equals">scopus</rel:condition>
    <rel:result>
      <rel:mask-field-value name="abstract"/>
    </rel:result>
  </rel:if>
</rel:inspect-each-object-record>
```

3.7 Inspecting objects

Objects can be inspected using the `<rel:inspect-object>` statement.

3.7.1 `<rel:inspect-object>`

The `<rel:inspect-object>` statement provides the ability to use conditional statements (e.g. if, choose) to inspect an object and select masks to apply to the object. This provides a flexible way to define masks, supporting much more functionality than is available using [Simple Mask Rules](#). (See [logic expressions](#) for full details of conditional logic statements.)

When a `<rel:condition>` evaluates as true, the associated `<rel:result>` is applied. A `<rel:inspect-object>` statement supports the following types of result mask statements:

- `<rel:mask-object>`
This masks the whole object, which deems the object as not relevant.
- `<rel:mask-field>`
This masks all occurrences of the specified field.

This example statement below masks all objects except journal-articles and conferences, and also masks the *author-url* and *keywords* fields.

```
<rel:inspect-object>
  <rel:if>
    <rel:condition operator="or">
      <rel:condition argument-field="object.type"
        ↪ operator="equals">journal-article</rel:condition>
      <rel:condition argument-field="object.type"
        ↪ operator="equals">conference</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:mask-field name="author-url"/>
      <rel:mask-field name="keywords"/>
    </rel:result>
  <rel:else>
    <rel:result>
      <rel:mask-object/>
    </rel:result>
  </rel:else>
</rel:if>
</rel:inspect-object>
```

Note that the same outcome could be achieved by using two simple rules in a mask:

```
<rel:exclude-objects-of-types-except>
  <rel:type name="journal-article"/>
  <rel:type name="conference"/>
```



```

</rel:exclude-objects-of-types-except>
<rel:exclude-fields>
  <rel:field name="author-url"/>
  <rel:field name="keywords"/>
</rel:exclude-fields>

```

3.8 Inspecting object records

The records of an object can be inspected using the `<rel:inspect-each-object-record>` statement.

3.8.1 `<rel:inspect-each-object-record>`

The `<rel:inspect-each-object-record>` statement provides the ability to use [logic expressions](#) (e.g. if, choose) to inspect each record of an object and select masks to apply to the object or record. This provides a flexible way to define masks, supporting much more functionality than is available using [Simple Mask Rules](#).

When a `<rel:condition>` evaluates as true, the associated `<rel:result>` is applied. An `<rel:inspect-each-object-record>` statement supports the following types of result mask statements:

- `<rel:mask-object>`
This masks the whole object, which deems the object as not relevant.
- `<rel:mask-record>`
This masks the current record being inspected. If all records for an object are masked, this deems the object as not relevant.
- `<rel:mask-field-value>`
This masks the specified field in the current record being inspected.

This example statement below masks all objects except journal-articles and conferences, and also masks the *abstract* in any scopus records.

```

<rel:inspect-each-object-record>
  <rel:if>
    <rel:condition operator="or">
      <rel:condition argument-field="object.type"
        ↪ operator="equals">journal-article</rel:condition>
      <rel:condition argument-field="object.type"
        ↪ operator="equals">conference</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:if>
        <rel:condition argument-field="record.source"
          ↪ operator="equals">scopus</rel:condition>

```

```

        <rel:result>
            <rel:mask-field-value name="abstract"/>
        </rel:result>
    </rel:if>
</rel:result>
<rel:else>
    <rel:result>
        <rel:mask-object/>
    </rel:result>
</rel:else>
</rel:if>
</rel:inspect-each-object-record>

```

Note that the same outcome could be achieved by using two simple rules in a mask:

```

<rel:exclude-objects-of-types-except>
    <rel:type name="journal-article"/>
    <rel:type name="conference"/>
</rel:exclude-objects-of-types-except>
<rel:exclude-source-fields>
    <rel:source-field source-name="scopus" field-name="abstract"/>
</rel:exclude-source-fields>

```

4 The `<rel:aggregations>` element

4.1 Introduction

The optional `<rel:aggregations>` element is used to define how records should be 'aggregated'. This is the process of obtaining a single value for a field from several values that may exist (each record may have a value for a field). Note that aggregation is performed after masking has been completed.

The approach taken is to define a source precedence, and calculate the aggregated field value as the field value of the highest precedence source which has a value.

If no `<rel:aggregations>` element exists, or no named `<rel:aggregation>` is specified by the `<rel:relevance-definition-selection>`, the default data source precedence is used. This can be viewed (and changed) on the data sources page (System Admin, Data Source Management - Data source precedences section).

The optional `<rel:aggregations>` element contains one or more named `<rel:aggregation>` elements. Each `<rel:aggregation>` element defines a source precedence.

4.2 `<rel:aggregation>` element

The `<rel:aggregation>` element defines a source precedence (using a `<rel:source-precedence>` element) to be used when aggregating the records of an object. Child `<rel:source>` elements define the source sequence, and can optionally be followed by a `<rel:other-default-precedence-sources>` element to indicate that all unreferenced sources should also be used, in the default data source precedence order.

Note that if only a subset of sources is referenced in an `<rel:aggregation>` element, any object which does not have any records from that subset will be treated as not relevant (so the aggregation can have the same effect as masking records).

The aggregation below uses the manual (verified) source as the top precedence source, then follows the default data source precedence for all other sources:

```
<rel:aggregation name="prefer-manual-verified-if-present">
  <rel:source-precedence>
    <rel:source name="manual" partition="verified" />
    <rel:other-default-precedence-sources />
  </rel:source-precedence>
</rel:aggregation>
```

This next example aggregation uses *dimensions* as the top precedence source, then *wos* and finally *scopus*. Records from all other data sources will be ignored (there is no `<rel:other-default-precedence-sources>` element present). Also note that if the object being processed

does not have any records from *dimensions*, *wos* or *scopus* (or those records have been masked) the object has no data to be aggregated and will be treated as not relevant:

```
<rel:aggregation name="use-dimension-wos-scopus-only">
  <rel:source-precedence>
    <rel:source name="dimensions"/>
    <rel:source name="wos"/>
    <rel:source name="scopus"/>
  </rel:source-precedence>
</rel:aggregation>
```

Note that when relevance schemes are being used for RT2 metadata updates you should always include the RT2 data source in your aggregation, but make sure it is not the top precedence data source. If it is the top precedence source, typically no metadata updates will be performed as the values selected to be updated will be the values that are already present in the repository. (For the same reason, if you are not using a defined aggregation, ensure that the default data source precedence does not have the repository as the top precedence data source.)

5 The `<rel:post-aggregation-masks>` element

5.1 Introduction

The optional `<rel:post-aggregation-masks>` element enables further masking to be performed on an object once aggregation has been performed. This may be required in some cases where the requirement for masking is driven by the value of a field. It is only after aggregation that a field has a single value - prior to aggregation it may have multiple values, up to one per record.

A `<rel:post-aggregation-masks>` element contains one or more uniquely named `<rel:post-aggregation-mask>` elements. Each `<rel:post-aggregation-mask>` element contains a set of inspection rules that are applied to an aggregated object, in sequence, to mask (hide) the object.

The name of the mask to be applied to an object is defined using the `<rel:relevance-definition-selector>` (see [Section 7](#)).

5.2 `<rel:post-aggregation-mask>` element

The `<rel:post-aggregation-mask>` element uses one or more `<rel:inspect-aggregated-object>` elements to inspect the aggregated object and decide whether to mask the object or not.

5.2.1 `<rel:inspect-aggregated-object>`

The `<rel:inspect-aggregated-object>` statement provides the ability to use [logic expressions](#) (e.g. if, choose) to inspect an aggregated object and select whether to mask the object.

When a `<rel:condition>` evaluates as true, the associated `<rel:result>` is applied. An `<rel:inspect-aggregated-object>` statement supports the following types of result mask statements:

- `<rel:mask-object>`
This masks the whole object, which deems the object as not relevant.

An example of a `<rel:post-aggregation-mask>` element is shown below. If there is no publication-date, the object is masked (and treated as not relevant):

```
<rel:post-aggregation-mask name="ignore-if-no-publication-date">
  <rel:inspect-aggregated-object>
    <rel:if>
      <rel:condition operator="not">
        <rel:condition argument-field="publication-date" operator="has-value"
          ↪ />
      </rel:condition>
    <rel:result>
```

```

        <rel:mask-object />
      </rel:result>
    </rel:if>
  </rel:inspect-aggregated-object>
</rel:post-aggregation-mask>

```

This example masks an object unless the publication date has a value which is on or after 1980-01-01:

```

<rel:post-aggregation-mask name="require-publication-date-from-1980">
  <rel:inspect-aggregated-object>
    <rel:choose>
      <rel:when>
        <rel:condition operator="not">
          <rel:condition argument-field="publication-date"
            ↪ operator="has-value" />
        </rel:condition>
        <rel:result>
          <rel:mask-object />
        </rel:result>
      </rel:when>
      <rel:when>
        <rel:condition argument-field="publication-date"
          ↪ operator="less-than">1980-01-01</rel:condition>
        <rel:result>
          <rel:mask-object />
        </rel:result>
      </rel:when>
    </rel:choose>
  </rel:inspect-aggregated-object>
</rel:post-aggregation-mask>

```

6 The `<rel:hash-definitions>` element

6.1 Introduction

The last stage of relevance assessment is to detect whether the data to be pushed to the external system has changed since the last push was made. This is performed by 'hashing' the values of all the fields that may be sent to the external system. If any of the field values have changed from the last time the hash was calculated, this will result in a different hash value from previously.

The `<rel:hash-definitions>` element allows one or more uniquely named `<rel:hash-definition>` elements to be defined. These specify which data values should be used when calculating the hash.

6.2 `<rel:hash-definition>`

The `<rel:hash-definition>` element identifies a set of data values which may be pushed to an external system. This set should include all the data values that could be pushed, but no more. The hashing mechanism will detect any change in the specified values, and initiate a re-push of the object.

Getting the correct set of values is important - if a value is missing from a hash definition, any change in that value will not cause the object to be re-pushed. Conversely, if a data value which is not used when pushing data is included in a hash definition, any change in that value will cause the object to be re-pushed needlessly. Any field which is referenced in a deposit crosswalk should be present in the hash definition.

An example hash definition is shown below:

```
<rel:hash-definition name="default">
  <rel:value from="authors">
    <!-- If more author data is being pushed, these values may need to be
    ↪ extended -->
    <rel:data-part name="person:lastname" />
    <rel:data-part name="person:firstnames" />
  </rel:value>
  <rel:value from="volume" />
  <rel:value from="issue" />
  <rel:value from="pagination" />
  <rel:value from="doi" />
  <rel:value from="acceptance-date" />
  <rel:value from="publication-date" />
  <rel:value from="online-publication-date" />
</rel:hash-definition>
```

The `<rel:value>` element identifies the data values to be hashed. The name of the value is

specified using the `from` attribute. These names can be any one of:

- [Object Property Names](#)
- [Record Property Names](#)
- [Record Field Names](#)
- [User Property Names](#) (when processing user objects only).

Some data values are complex values made up from parts (e.g. the *authors* field has a type of person-list, and contains firstnames, lastnames etc). In order to avoid having to always hash all parts it is possible to specify which parts to use in a hash by optionally adding child `<rel:data-part>` elements (which themselves may contain `<rel:data-part>` elements...). This improves efficiency and avoids redundant pushes to the external system. See [data-part Names](#) for details.

6.2.1 Hashing link data - `<rel:link-types>` element

As well as hashing object properties it is sometimes important to hash links, because link data may be pushed in some situations.

The optional `<rel:link-types>` element may be specified following all `<rel:value>` elements. This contains one or more `<rel:link-type>` elements, whose `name` specifies the name of a link type which is to be hashed. This will hash the details of all approved links of the given type, and will detect any changes in these links.

Commonly used link types include:

Link Type Name	Description
publication-user-authorship	Link between a publication and an author
publication-grant-funded	Link between a publication and a grant

7 The `<rel:relevance-definition-selector>` element

7.1 Introduction

A relevance definition file may contain several definitions for masks, aggregations, post aggregation masks and hash definitions. When an object is assessed for relevance the `<rel:relevance-definition-selector>` element is used to select which mask, aggregation, post aggregation mask and hash definition are to be used.

These are specified using a `<rel:relevance-definition-selection>` element. If no `<rel:relevance-definition-selection>` element results from evaluating the `<rel:relevance-definition-selector>` element, the object is treated as not relevant.

The `<rel:relevance-definition-selector>` statement provides the ability to use [logic expressions](#) (e.g. `if`, `choose`) to inspect an object. This is performed as the first stage of the relevance assessment, before any aggregation has been performed, so conditions may only reference object properties (see [Object Property Names](#)).

When a `<rel:condition>` evaluates as true, the associated `<rel:result>` is applied, and may return a `<rel:relevance-definition-selection>` element.

7.2 `<rel:relevance-definition-selector>`

The `<rel:relevance-definition-selector>` element is used to select a `<rel:relevance-definition-selection>` element for an object that is to be assessed for relevance.

For simple cases where no conditional logic is required, a single `<rel:relevance-definition-selection>` element is all that is required to specify how relevance is to be assessed.

The example below shows the simplest possible `<rel:relevance-definition-selection>` element, which just specifies a single hash definition to be used for all objects (which will all be treated as relevant, because there is no logic selection or masking):

```
<rel:relevance-definition-selector>
  <rel:relevance-definition-selection hash-definition="default" />
</rel:relevance-definition-selector>
```

This example shows another simple `<rel:relevance-definition-selection>` element, which specifies which mask, aggregation, post-aggregation mask and hash definition should be used:

```
<rel:relevance-definition-selector>
  <rel:relevance-definition-selection
    mask="only-use-journal-articles"
    aggregation="prefer-manual-verified-if-present"
    post-aggregation-mask="require-publication-date-from-1980"
```

```

        hash-definition="hash-for-journal-articles" />
</rel:relevance-definition-selector>

```

Logic statements can be used to support different behaviours for different types of object. This example uses a choose statement to specify different `<rel:relevance-definition-selection>` elements for journal articles and conferences, and nothing for other publication types, which will be treated as not relevant.

```

<rel:relevance-definition-selector>
  <rel:choose>
    <rel:when>
      <rel:condition argument-field="object.type"
        ↪ operator="equals">journal-article</rel:condition>
      <rel:result>
        <rel:relevance-definition-selection
          mask="ensure-licensing-adhered-to"
          aggregation="prefer-manual-verified-if-present"
          ↪ post-aggregation-mask="require-publication-date-from-1980"
          ↪ hash-definition="journal-article-hash" />
        </rel:result>
      </rel:when>
      <rel:when>
        <rel:condition argument-field="object.type"
          ↪ operator="equals">conference</rel:condition>
        <rel:result>
          <rel:relevance-definition-selection
            mask="ensure-licensing-adhered-to"
            hash-definition="conference-hash" />
          </rel:result>
        </rel:when>
      </rel:choose>
    </rel:relevance-definition-selector>

```

7.2.1 `<rel:relevance-definition-selection>`

The `<rel:relevance-definition-selection>` identifies how an object is to be assessed for relevance.

The following attributes are used:

- `mask` attribute (optional)
Specifies the name of the mask to be applied to an object. The attribute value must match the name of a mask defined in the `<rel:masks>` element. If no `mask` attribute is specified, no mask will be applied during relevance assessment.
- `aggregation` attribute (optional)
Specifies how record field values are to be aggregated. The attribute value must match

the name of an aggregation defined in the `<rel:aggregations>` element. If no aggregation attribute is specified, aggregation will be performed using the default data source precedence.

- `post-aggregation-mask` attribute (optional)
Specifies additional masking to be performed after aggregation has been performed. The attribute value must match the name of a post-aggregation mask defined in the `<rel:post-aggregation-masks>` element. If no `post-aggregation-mask` attribute is specified, no post-aggregation mask will be applied.
- `hash-definition` attribute (required)
Specifies the hash definition to be used if the object being assessed is found to be relevant. The attribute value must match the name of a hash definition defined in the `<rel:hash-definitions>` element.

A Logic Expressions

Often the way in which relevance is assessed will depend on the object that is being processed. For instance, a publication may be relevant if it is of a certain type, if it has a publication date after 2000 or if it has a record from one or more specific sources. Each of these require the ability to specify and evaluate logic statements to provide the required functionality.

Relevance schemes support logic expressions with **if** statements (using an `<rel:if>` element) and **choose** statements (using the `<rel:choose>` element). A logic expression uses a `<rel:condition>` element to define a condition to be evaluated, and a `<rel:result>` element to hold the result to be used when the condition is true.

Logic expressions can be used within the following relevance scheme elements:

- `<rel:inspect-object>`
- `<rel:inspect-each-object-record>`
- `<rel:inspect-aggregated-object>`
- `<rel:relevance-definition-selector>`.

A.1 Conditions - the `<rel:condition>` element

Logic conditions are defined using `<rel:condition>` elements. Each `<rel:condition>` element is evaluated to either true or false.

A single `<rel:condition>` element can be used to define a simple single statement (e.g. is this object type 'journal-article'), or can be combined with other child conditions (using AND, OR and NOT logic operators) to provide more complex logic.

A.1.1 Condition data types

Relevance scheme conditions support the following data types:

- **boolean**
Supports values of *true* or *false*.

When used with the *equals* or *not-equals* operator, the following condition values are supported:
 - true, TRUE, 1 (for true)
 - false, FALSE, 0 (for false).
- **string**
A sequence of text characters.
- **string list**
A list of zero or more strings.

When used with the *contains-any-of* or *contains-all-of* operator, the `condition` value consists of comma-separated strings (e.g. "red,green,blue").

- **date**

A date, matches the dates used within Elements (i.e. precision may vary - year, month or day).

When used with a `condition`, a date value should use one of the following formats:

- 2017 (year precision - YYYY)
- 2017-05 (month precision - YYYY-MM)
- 2017-05-23 (day precision - YYYY-MM-DD).

- **timestamp**

A point in time, including date and time.

When used with a `condition`, a timestamp value should always include a year, month and day, time component is optional:

- 2014-09-16 (YYYY-MM-DD, time defaults to 00:00:00)
- 2014-09-16 16:11:12 (YYYY-MM-DD hh:mm:ss).

- **number**

A numeric value.

- **other**

A type which is none of the above.

A.1.2 Condition operators - the `operator` attribute

A `<rel:condition>` element must have one of the following values for the `operator` attribute:

Option	Description
<code>equals</code>	Performs a comparison between the property referenced by the <code>argument-field</code> attribute and the specified value. Returns <i>true</i> if they are the same, otherwise <i>false</i> .
<code>not-equals</code>	Applies the logical NOT operator to the result of an <code>equals</code> operation (see above).
<code>contains-substring</code>	Inspects the string referenced by the <code>argument-field</code> attribute and determines whether it contains the specified value.
<code>contains-any-of</code>	Inspects the list of strings referenced by the <code>argument-field</code> attribute and determines whether it contains any of the specified list of strings, which is represented by values separated by a comma (,).
<code>contains-all-of</code>	Inspects the list of strings referenced by the <code>argument-field</code> attribute and determines whether it contains all of the specified list of strings, which is represented by values separated by a comma (,).
<code>has-value</code>	Returns <i>true</i> if the property referenced by the <code>argument-field</code> attribute has a value, otherwise <i>false</i> .
<code>greater-than</code>	Inspects the property referenced by the <code>argument-field</code> attribute and determines whether it is greater than the specified value or not.
<code>greater-than-or-equals</code>	Inspects the property referenced by the <code>argument-field</code> attribute and determines whether it is greater than or equals the specified value or not.
<code>less-than</code>	Inspects the property referenced by the <code>argument-field</code> attribute and determines whether it is less than the specified value or not.
<code>less-than-or-equals</code>	Inspects the property referenced by the <code>argument-field</code> attribute and determines whether it is less than or equals the specified value or not.
<code>not</code>	Applies the logical NOT operator to the result of the child <code><rel:condition></code> element.
<code>and</code>	Applies the logical AND operator to the result of all child <code><rel:condition></code> elements.
<code>or</code>	Applies the logical OR operator to the result of all child <code><rel:condition></code> elements.

The `not`, `and` and `or` operators provide a means to build more complex conditional statements.

A.1.3 Condition operator support by data type

Condition operators typically support a limited range of data types. The table below shows which operators apply to which data types:

Condition data type	equals	not-equals	contains-substring	contains-any-of	contains-all-of	has-value	greater-than	greater-than-or-equals	less-than	less-than-or-equals
boolean	X	X				X				
string	X	X	X			X				
string list				X	X	X				
date	X	X				X	X	X	X	X
timestamp	X	X				X	X	X	X	X
number	X	X				X	X	X	X	X
other						X				

An operator must not be applied to an inappropriate data type.

Special consideration is needed for the behaviour of date data types. These types can be defined with a precision of year (e.g. 2018), month (e.g. 2018-04) or day (e.g. 2018-04-21). When comparing two dates the highest common precision is found, and values are compared using that precision.

A.1.4 Mapping Elements data types to Condition data types

Elements supports a much richer set of data types than conditions, so in order to be able to use Elements data types in conditions they need to be mapped to a suitable condition data type. These mappings are shown in the table below:

Elements data type	Condition data type	Description
academic-appointment-list	other	
address-list	other	
boolean	boolean	
certification-list	other	
choice	string	
comment-list	string list	
date	date	
degree-list	other	
doi	string	
email-address-list	other	
funding-acknowledgements	other	
identifier-list	other	
integer	number	
isbn-10	string	
isbn-13	string	
issn	string	
issn-list	string list	
keyword-list	string list	
language-competency-list	other	
list	string list	
money	number	
non-academic-employment-list	other	
number	number	
pagination	other	
person	other	
person-list	other	
phone-number-list	other	
postgraduate-training-list	other	
text	string	
url	string	
web-address-list	other	

Thus a “doi” field, for instance, is treated as a string in conditions, and can be used with the *equals*, *not-equals*, *contains-substring* and *has-value* operators.

A.2 Results - the `<rel:result>` element

When a logic expression condition evaluates to *true*, the associated `<rel:result>` element is evaluated.

The allowed contents of a `<rel:result>` element depends on its context, but may always contain further logic expressions (i.e. logic expressions may be nested).

In addition to further logic expressions (if, choose), the following elements are supported, based on context:

- within a `<rel:inspect-object>` element
 - `<rel:mask-object>` element and
 - `<rel:mask-field>` elements
- within a `<rel:inspect-each-object-record>` element
 - `<rel:mask-object>` element,
 - `<rel:mask-record>` element and
 - `<rel:mask-field-value>` elements
- within a `<rel:inspect-aggregated-object>` element
 - `<rel:mask-object>` element
- within a `<rel:relevance-definition-selector>` element
 - `<rel:relevance-definition-selection>` element

A.3 The `<rel:if>` element

The `<rel:if>` element contains:

- a `<rel:condition>` element
 - which defines a single condition, which may evaluate to *true* or *false*
- a `<rel:result>` element
 - which is evaluated if the result of the `<rel:condition>` element is true
- optionally, an `<rel:else>` element
 - which is evaluated if the result of the `<rel:condition>` element is false.

The following example uses the `not-equals` condition operator to mask any object which does not have a type of 'journal-article'.

```
<rel:inspect-object>
  <rel:if>
    <rel:condition argument-field="object.type"
      ↪ operator="not-equals">journal-article</rel:condition>
    <rel:result>
      <rel:mask-object />
    </rel:result>
```

```

    </rel:if>
</rel:inspect-object>

```

A `<rel:if>` element can also contain a `<rel:else>` element; if the condition within the `<rel:if>` is not true, the result of the `<rel:else>` element is used. The following example, based on the one above, additionally includes an `<rel:else>` element to mask all values in the custom field 'c-my-private-field'. The `<rel:else>` mask will only be applied to objects which don't match the `<rel:if>` condition, i.e. objects which are of type 'journal-article':

```

<rel:inspect-object>
  <rel:if>
    <rel:condition argument-field="object.type"
      ↪ operator="not-equals">journal-article</rel:condition>
    <rel:result>
      <rel:mask-object />
    </rel:result>
    <rel:else>
      <rel:result>
        <rel:mask-field name="c-my-private-field"/>
      </rel:result>
    </rel:else>
  </rel:if>
</rel:inspect-object>

```

The example below shows an `<rel:condition>` using the `and` operator to combine the results of other child conditions. It returns true only if `object.type` is equal to "journal-article" and `object.approved-link-types` contains "publication-user-authorship". The effect is that the only objects that are not masked are journal articles which have at least one authorship link:

```

<rel:inspect-object>
  <rel:if>
    <rel:condition operator="and">
      <rel:condition argument-field="object.type"
        ↪ operator="not-equals">journal-article</rel:condition>
      <rel:condition argument-field="object.approved-link-types"
        ↪ operator="contains-all-of">publication-user-authorship</rel:condition>
    </rel:condition>
    <rel:result>
      <rel:mask-object />
    </rel:result>
  </rel:if>
</rel:inspect-object>

```

A.4 The `<rel:choose>` element

The `<rel:choose>` element can be used to evaluate a number of conditions in sequence, and return the result of the first condition which is true.

The `<rel:choose>` element contains:

- one or more `<rel:when>` elements
each of which defines a single condition, which may evaluate to *true* or *false* and a `<rel:result>` element which is evaluated if the result of the `<rel:condition>` element is *true*
- optionally, an `<rel:otherwise>` element
which is evaluated if none of the previous `<rel:when>` element conditions have evaluated to *true*.

The `<rel:choose>` element uses the same `<rel:condition>` and `<rel:result>` elements that are used by the `<rel:if>` element. The use of multiple `<rel:when>` elements makes it more flexible than the `<rel:if>` element in some circumstances.

The example choose expression below is used within an `<rel:inspect-object>` element:

- The **first** `<rel:when>` **statement** masks an object if it is not a journal article or a conference,
- The **second** `<rel:when>` **statement** masks an object if it is not related to groups with IDs 12, 45 or 67,
- The **otherwise statement** masks the abstract field for an object if it does not match either of the previous `<rel:when>` statements.

```
<rel:inspect-object>
  <rel:choose>
    <rel:when>
      <!-- Mask all objects which are not journal articles or conferences -->
      <rel:condition operator="not">
        <rel:condition operator="or">
          <rel:condition argument-field="object.type"
            ↪ operator="equals">journal-article</rel:condition>
          <rel:condition argument-field="object.type"
            ↪ operator="equals">conference</rel:condition>
        </rel:condition>
      </rel:condition>
      <rel:result>
        <rel:mask-object/>
      </rel:result>
    </rel:when>
    <rel:when>
      <!-- Mask all objects not related to groups with ids 12, 45 or 67 -->
      <rel:condition operator="not">
        <rel:condition argument-field="object.group-ids"
          ↪ operator="contains-any-of">12,45,67</rel:condition>
```

```
</rel:condition>
<rel:result>
  <rel:mask-object/>
</rel:result>
</rel:when>
<rel:otherwise>
  <!-- Otherwise, mask the abstract field -->
  <!-- Note: This will only apply to journal articles or conferences which are
  ↳ related to groups 12, 45 or 67 -->
  <rel:result>
    <rel:mask-field name="abstract"/>
  </rel:result>
</rel:otherwise>
</rel:choose>
</rel:inspect-object>
```

B Property Names and Data-parts

Relevance scheme files reference data properties using names. These are documented below.

B.1 Object Property Names

Property Name	Data Type	Description
object.id	Number	The Elements internal identifier for this object.
object.category	String	The name of the category to which this object belongs.
object.type	String	The name of this object's type.
object.created-when	Timestamp	The date and time when this object was created in Elements.
object.last-modified-date	Timestamp	The date and time when this object was last modified in Elements.
object.last-affected-date	Timestamp	The date and time when this object, one of its relationships or a related object was last modified in Elements.
object.reporting-date-1	Timestamp	The value of reporting-date-1 for this object.
object.reporting-date-2	Timestamp	The value of reporting-date-2 for this object.
object.labels	StringList	A list of all the labels for this object.
object.group-ids	StringList	A list of the IDs of all the groups related to this object, including parent groups.
object.explicit-group-ids	StringList	A list of the IDs of all the groups explicitly related to this object.
object.record-sources	StringList	A list of the names of all the record sources of this object.
object.approved-link-types	StringList	A list of the names of all approved link types of this object.

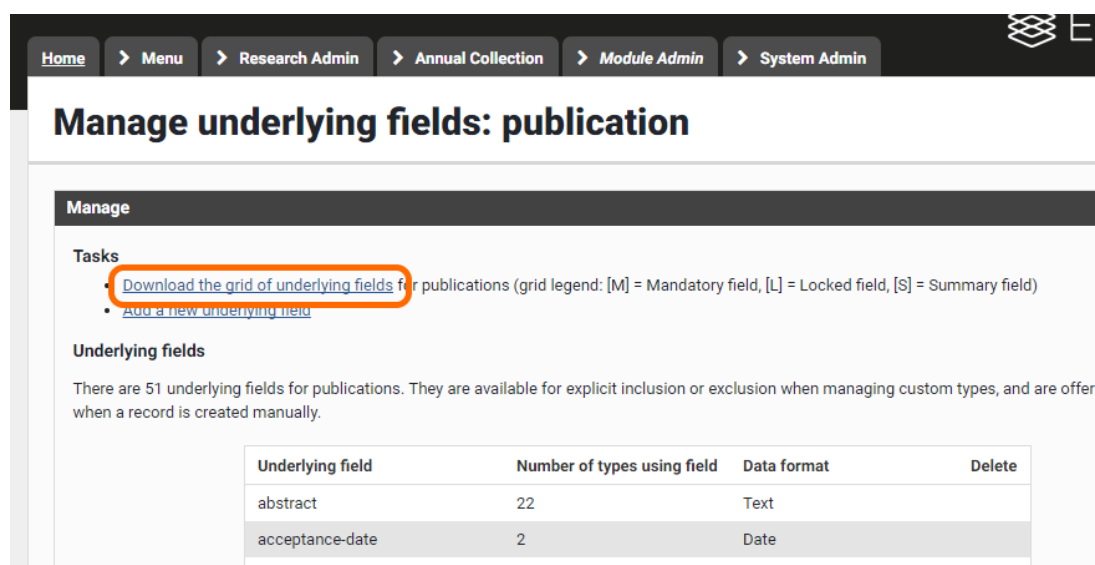
B.2 Record Property Names

Property Name	Data Type	Description
record.id	Number	The Elements internal identifier for this record.
record.proprietary-id	String	The proprietary-id for this record (which represents a unique identifier for the record in its source system).
record.source	String	The name of the source from which this record comes.

B.3 Record Field Names

Elements uses a standard set of field names for each category, and supports the addition of custom fields (whose names all start 'c-'). All these fields are available for use in relevance schemes.

For a full list of all the fields and their usages, navigate to Module Admin, *Category*, Underlying Fields, then click on Download the grid of underlying fields.



Manage underlying fields: publication

Manage

Tasks

- [Download the grid of underlying fields](#) for publications (grid legend: [M] = Mandatory field, [L] = Locked field, [S] = Summary field)
- [Add a new underlying field](#)

Underlying fields

There are 51 underlying fields for publications. They are available for explicit inclusion or exclusion when managing custom types, and are offer when a record is created manually.

Underlying field	Number of types using field	Data format	Delete
abstract	22	Text	
acceptance-date	2	Date	

B.3.1 Publication field Names

This table shows the standard fields names used for publications. In addition to these names, you are able to use the names of any custom fields you define.

Field Name	Data Type	Description
abstract	String	
acceptance-date	Date	
addresses	address-list	
altmetric-attention-score	Number	
are-files-confidential	Bool	
associated-authors	person-list	
associated-identifiers	identifier-list	
author-licence	String	
authors	person-list	
author-url	String	
collections	StringListist	
commissioning-body	String	
confidential	Bool	
confidential-files-reason	String	
doi	string	
edition	String	

Field Name	Data Type	Description
editors	person-list	
eissn	String	
embargo-release-date	Date	
external-identifiers	identifier-list	
field-citation-ratio	Number	
filed-date	Date	
finish-date	Date	
funding-acknowledgements	funding-acknowledgements	
isbn-10	String	
isbn-13	String	
is-compliant-with-inst-policy	Bool	
is-embargoed	Bool	
is-open-access	Bool	
issn	String	
issue	String	
journal	String	
keywords	keyword-list	
language	String	
location	String	
medium	String	
name-of-conference	String	
notes	String	
number	String	
number-of-pieces	String	
oa-location-file-version	String	
oa-location-url	String	
online-publication-date	Date	
pagination	pagination	
parent-title	String	
patent-number	String	
patent-status	String	
pii	String	
place-of-publication	String	
publication-date	Date	
publication-status	choice	
public-url	String	
publisher	String	
publisher-licence	choice	
publisher-url	String	

Field Name	Data Type	Description
record-created-at-source-date	Date	
record-made-public-at-source-date	Date	
references	StringList	
relative-citation-ratio	number	
repository-status	String	
series	String	
start-date	Date	
thesis-type	String	
title	String	
types	StringList	
version	String	
volume	String	

B.4 data-part Names

data-part Name	Data Type	Description
address:city	string	The city part of an address
address:country	string	The country part of an address
address:full	string	The full address as a single string
address:iso-country-code	string	The ISO country code of an address
address:name	string	The name of an address
address:organisation	string	The organisation part of an address
address:state	string	The State of an address
address:streetaddress	string	The street part of an address
address:suborganisation	string	The sub-organisation part of an address
address:type	string	The type of an address
address:zipcode	string	The zipcode of an address
certification:description	string	The description of a certification
certification:effective-date	date	The effective date of a certification
certification:expiry-date	date	The expiry date of a certification
certification:institution	string	The institution that awarded a certification
certification:title	string	The title of a certification
date:day	string	The day of the month part of a date
date:month	string	The month part of a date (1 to 12)
date:precision	string	The precision part of a date
date:year	string	The year part of a date

data-part Name	Data Type	Description
degree:end-date	date	The end-date part of a degree
degree:field-of-study	string	The field of study of a degree
degree:institution	string	The institution that awarded a degree
degree:name	string	The name of a degree
degree:start-date	date	The start date of a degree
degree:supervisors	person-list	The supervisors of a degree
degree:thesis	string	The thesis title of a degree
identifier:scheme	string	The name of an identifier scheme (may be empty)
identifier:value	string	The value of an identifier
keyword:percent	number	The percentage weighting associated with a label
keyword:scheme	string	The name of a keyword scheme (may be empty)
keyword:value	string	The value of a keyword
money:amount	number	The amount of a money value
money:currency	string	The currency of a money value
pagination:begin-page	string	The start page
pagination:end-page	string	The end page
pagination:page-count	string	The page count
person-role:type	string	The type of a person role
person-role:value	string	The value of a person role
person:address	address	The address of a person
person:email-address	string	The email address of a person
person:firstnames	string	The first names of a person
person:id	string	The internal Elements ID of a person
person:identifier	string	The identifiers associated with a person
person:initials	string	The initials of a person
person:lastname	string	The last name of a person
person:role	role	The role of a person
web-address:label	string	The label to be used with a web address
web-address:type	string	The type of a web address
web-address:url	string	The url of a web address

B.5 User Property Names

data-part Name	Data Type	Description
Property Name	Data Type	Description
user.id	Number	The Elements internal identifier for this user.
user.proprietary-id	String	The proprietary-id for this user (which represents a unique identifier for the user in its source system).
user.username	String	The username of this user (as used during Elements login).
user.initials	String	The initials of this user.

data-part Name	Data Type	Description
user.first-name	String	The first name of this user.
user.last-name	String	The last name of this user.
user.email-address	String	The email address of this user.
user.institutional-email-is-public	Bool	Whether this user's email address is public.
user.primary-group-descriptor	String	This user's primary group descriptor.
user.is-public	Bool	Whether this user data is publicly accessible.
user.is-current-staff	Bool	Whether this user is a current member of staff.
user.is-academic	Bool	Whether this user is an academic member of staff.
user.is-login-allowed	Bool	Whether this user is allowed to login to Elements.
user.title	String	This user's title.
user.position	String	This user's position.
user.department	String	This user's department.
user.public-url-path-fragment	String	This user's url path fragment (may be used for providing a URL for a user's profile in some systems).
user.claimed	Bool	Whether this user has been marked as relevant for the institution.
user.photo-last-modified-date	DateTime	The date and time this user's photo was last modified.
user.preferred-first-name	String	This user's preferred first name.
user.preferred-last-name	String	This user's preferred last name.
user.arrive-date	String	This user's arrive date.
user.leave-date	String	This user's leave date.
user.known-as	String	This user's known-as name.
user.suffix	String	This user's name suffix.
user.generic1	String	This user's generic field 1.
user.generic2	String	This user's generic field 2.
user.generic3	String	This user's generic field 3.
user.generic4	String	This user's generic field 4.
user.generic5	String	This user's generic field 5.
user.generic6	String	This user's generic field 6.
user.generic7	String	This user's generic field 7.
user.generic8	String	This user's generic field 8.
user.generic9	String	This user's generic field 9.
user.generic10	String	This user's generic field 10.

C Alternative File Structure (Obsolete)

An alternative structure has been used in the past for relevance schemes. While this continues to be supported, it should be considered obsolete and no new relevance scheme files should use this structure.

The alternative structure supports much less functionality than the full structure described previously. In particular it only supports a single mechanism to assess relevance and detect changes for each category, and there is no support for generic logic conditions.

C.1 Components of alternative file structure

The structure of a relevance scheme file using the alternative structure is shown below.

```
<xml-relevance-scheme
  xmlns="http://www.symplectic.co.uk/elements/xml-relevance-scheme">

  <!-- required element for each relevant category. -->
  <category name="publication">

    <mask>
      <!-- Optional element. Contains a mask to be applied to the object. -->
    </mask>

    <aggregation>
      <!-- Optional element. Contains an aggregation. -->
    </aggregation>

    <relevance-conditions>
      <!-- Required element. Contains statements that specify which objects are to be
      ↪ considered relevant. -->
    </relevance-conditions>

    <hash-definition>
      <!-- Required element. Contains a single hash definition. -->
    </hash-definition>

  </category>
</xml-relevance-scheme>
```

D Best Practices when writing relevance schemes

A relevance scheme may be a complex document, that needs to be maintained accurately to ensure that the correct data is pushed to external systems.

The following guidelines are suggested as best practice to ensure that your relevance schemes are understandable and maintainable:

- [Design for maintenance](#)
- [Design for performance](#).

D.1 Design for maintenance

A significant effort is often required to prepare a relevance definition, which can involve obtaining a detailed understanding of operational practices and procedures. At a later date changes to practices and procedures will require changes to relevance schemes. In order to ensure the process of maintaining relevance schemes is as simple as possible, and that knowledge of the required behaviours is not lost to the organisation, the following practices are recommended.

D.1.1 Keep things simple

The relevance scheme syntax may often provide several possible ways to achieve the same result, especially when masking objects. In general, use a simple mechanism rather than a complex one.

The example statements below mask all objects except journal-articles and conferences, and also masks the *abstract* in any scopus records. The first is simpler to understand and maintain.

Prefer:

```
<rel:exclude-objects-of-types-except>
  <rel:type name="journal-article"/>
  <rel:type name="conference"/>
</rel:exclude-objects-of-types-except>
<rel:exclude-source-fields>
  <rel:source-field source-name="scopus" field-name="abstract"/>
</rel:exclude-source-fields>
```

to:

```
<rel:inspect-each-object-record>
  <rel:if>
    <rel:condition operator="or">
      <rel:condition argument-field="object.type"
        ↪ operator="equals">journal-article</rel:condition>
      <rel:condition argument-field="object.type"
        ↪ operator="equals">conference</rel:condition>
```

```

    </rel:condition>
    <rel:result>
      <rel:if>
        <rel:condition argument-field="record.source"
          ↪ operator="equals">scopus</rel:condition>
        <rel:result>
          <rel:mask-field-value name="abstract"/>
        </rel:result>
      </rel:if>
    </rel:result>
    <rel:else>
      <rel:result>
        <rel:mask-object/>
      </rel:result>
    </rel:else>
  </rel:if>
</rel:inspect-each-object-record>

```

D.1.2 Use meaningful names

Most items defined within a relevance scheme have names. These are used to uniquely identify items. In order to help understanding and assist maintenance we recommend that you use names that are meaningful, and describe the functionality that is supported:

Prefer:

```
<rel:mask name="mask-for-journal-articles">
```

to:

```
<rel:mask name="mask-01">
```

D.1.3 Use comments

The addition of comments to a relevance scheme can make it much easier to understand and maintain. As relevance schemes are defined using an XML format, comment nodes may be added anywhere without impacting the operation of relevance processing. Prefer:

```

<!--
  Behaviour: Ensure only journal articles and conferences pushed to repository.
  Reason: Other types should not be put into the repository.
-->
<rel:exclude-objects-of-types-except>
  <rel:type name="journal-article"/>

```

```

    <rel:type name="conference"/>
</rel:exclude-objects-of-types-except>
<!--
    Behaviour: Mask Abstract field from Scopus records.
    Reason: This is not licenced to be used outside the institution.
-->
<rel:exclude-source-fields>
    <rel:source-field source-name="scopus" field-name="abstract"/>
</rel:exclude-source-fields>

```

to:

```

<rel:exclude-objects-of-types-except>
    <rel:type name="journal-article"/>
    <rel:type name="conference"/>
</rel:exclude-objects-of-types-except>
<rel:exclude-source-fields>
    <rel:source-field source-name="scopus" field-name="abstract"/>
</rel:exclude-source-fields>

```

D.2 Design for performance

The flexible nature of the relevance scheme design means that it is possible to use several different means to achieve the same result, especially when masking data. However, some approaches may use more resources than others. The following steps can be taken to minimise resource usage when assessing.

D.2.1 Use `<masks>` in preference to `<post-aggregation-masks>`

Relevance assessment is performed as a sequence of steps: mask - aggregate - aggregated-mask - hash. By masking data as early as possible in this pipeline, the resources required to assess relevance can be minimised.

In particular, if an object is not relevant, try to identify this as early as possible, avoiding any further processing.

D.2.2 Avoid `<inspect-each-object-record>` if possible

The `<inspect-each-object-record>` operation is invoked once for each record of an object, while other operations are typically invoked once per object. Consider using `<inspect-object>`, `<rel:exclude-sources>`, `<rel:exclude-sources-except>` or `<rel:exclude-source-fields>` in preference.

D.2.3 Do not include excess fields in hash definitions

Change detection is performed based on the hash definition that is selected for an object. If a hash definition references any properties which are not used when pushing the object, any changes to these properties will cause redundant push operations to be performed. These should be avoided by removing any references to properties which are never pushed.