ACHeck21 Search Language

January 1

2015

This file describes search language used by all of the services provided by the ACHeck21 Gateway.

Table of Contents

Revision History

03/28/2010	CDW	Removed from global gateway document into its own document.
08/20/2010	CDW	Fixed explanation of REST queries. Text said values were separated by colons, example said it was commas. Example was right.

Documentation Suggestions

At ACHeck21, our goal is to provide payment processing solutions that help to make your business a success. If you have trouble with this document, or have suggestions for improvements, please email them to documentation@diversifiedchecksolutions.com. Please reference the publication date from the title page in any correspondence about the documentation.

Search / Query Language

Introduction

Many of the collection entities in the web services provide query operations. These queries are contained in "untyped" XML strings inside of the service request. By untyped, we mean that the internal structure of these queries is not specified in the web service description. This structure allows ACHeck21® to enhance the queries without invalidating any proxy stubs generated from the web service's WSDL.

Basic principles

The *Query* element consists of a sequence of named entities, each corresponding to a particular search criterion. The name of the element is the name of the property to be searched (each query-able element contains a list of searchable properties). The content of the element specifies the structure of the search using 2 tags: the *Operation* tag that defines the type of search (see below), and one or more *Value* tags to define the values to search for. The interpretation and number of the *Value* tags depends on the specific *Operation* tag. These semantics are described in detail in the remainder of this document.

There is no limit on the number of restrictions that can be placed on the search. In fact, the system will permit multiple, mutually-exclusive restrictions; the service will simply be an empty result set.

As of this writing, the only way to perform an *OR*-type search is by using an *IN* query. If the query cannot be sufficiently restricted using *IN* queries, then the calling application will need to perform post-processing on the returned values to get the desired results.

Operations

The following list describes the currently supported operations, and defines the number and interpretation of the *Value* tags. All operation names are case-insensitive.

Equal
NotEqual
LessThan
LessThanOrEqual
GreaterThan
GreaterThanOrEqual

There must be a single *Value* that defines the comparison. For example, the following element defines a query that searches for items have an *Amount* equal to 0:

```
<Amount>
  <Operation>Equal</Operation>
  <Value>0</Value>
</Amount>
```

Between

There must be two *Value* tags that define the (inclusive) range of values to search. For example, the following element defines a query that searches for items have an *Amount* between 0 and 100:

```
<Amount>
  <Operation>Between</Operation>
  <Value>0</Value>
  <Value>100</Value>
</Amount>
```

In

There must be one or more *Value* tags that define list of values to search. For example, the following element defines a query that searches for items have an *Amount* of 10, 20, or 30:

```
<Amount>
  <Operation>IN</Operation>
  <Value>10</Value>
  <Value>20</Value>
  <Value>30</Value>
</Amount>
```

Begins

There must be a single *Value* tags that gives the textual prefix to search for. For example, the following element defines a query that searches for items where the *Name* begins with “Mc”:

```
<Name>
  <Operation>Begins</Operation>
  <Value>Mc</Value>
</Name>
```

Ends

There must be a single *Value* tags that gives the textual suffix to search for. For example, the following element defines a query that searches for items where the *Name* ends with “son”:

```
<Name>
  <Operation>Ends</Operation>
```

```
  <Value>son</Value>
</Name>
```

Contains

There must be a single *Value* tags that gives the textual value to search for. For example, the following element defines a query that searches for items where the *Name* contains the text "donald":

```
<Name>
  <Operation>Contains</Operation>
  <Value>donald</Value>
</Name>
```

Like

This operation is the same as the SQL LIKE operator. The query must contain the % wildcard characters explicitly. There must be a single *Value* tag that gives the search pattern. For example, the following element defines a query that searches for items where the *Name* contains the text "donald" (this is the same as a *Contains* query):

```
<Name>
  <Operation>Like</Operation>
  <Value>%donald%</Value>
</Name>
```

Data value specifications

Most values are specified explicitly inside the *Value* tag in exactly the form that would be expected. There are a couple of things to be aware of. First, string values should not be enclosed in quotation marks. The services are able to determine the data types of the underlying system, and will add quotation marks as needed. Second, date values should be specified using the SQL standard format of *yyyy-mm-dd*. Failure to conform to this format will result in an error. Third, amount values should be specified without a dollar sign or any internal commas.

Query string format

When using the REST services, the query should be expressed in the request:

https://gateway.acheck21.com/GlobalGateway/REST/*entity*? *actual_query*

There are 2 formats for the actual query. First, it is possible to simply place the XML into the query string as the value of an item named "query". Alternately, and more HTTP-like, the query can be expressed as separate items on the query string, each element having the name of the item to be searched for, with the value of the item being the operation, followed by one or more values, all separated by commas. So, to search for batches for client 999999999 uploaded between 1-1-2008 and 1-31-2008, GET from the following URI:

/REST/batches/9999999999?UploadDate=between,2008-01-01,2008-01-31

To query for all batches uploaded in January 2008 with a status of *Deleted*, use the following URI:

/REST/batches/9999999999?UploadDate=between,2008-01-01,2008-01-31?Status=equal,Deleted