Heliophysics Integrated Observatory

Project No.: 238969
Call: FP7-INFRA-2008-2

Semantic Mapping Service User Guide

Version 0.1

Title:	**Semantic Mapping Service – User Guide**
Document No.:	HELIO-UNIMAN_S2_004_UM_SMS
Date:	21 March 2012
Editor:	**Anja Le Blanc**, The University of Manchester
Contributors:	
Distribution:	Project

Revision History

Version	Date	Released by	Detail
0.1	21/03/2012	Anja Le Blanc	Initial Draft

Note: Any notes here.

1 Introduction

The Semantic Mapping Service (SMS) provides a set of functions that use the HELIO ontology as a source of information. It is only implemented with a SOAP web service. Current locations of instances of this service can be found in the HELIO registry.

2 Use

The interface to this service is SOAP only. All programs or programming environments which support this protocol will have no problems accessing its functions. SMS is providing support functionality which is most useful in connection with other HELIO services.

2.1 Workflow environments

All workflow environments support SOAP. By providing the URL to the WSDL file of the service the workflow environment can read and understand the functions provided and the definition of their inputs and outputs.

2.2 Programming environments

Most programming environments provide libraries to handle SOAP interactions. Please refer to the documentation of these libraries for details.

2.2.1 JAVA

Java provides a tool ‘wsdl2java’ to generate a client API from a WSDL description

2.2.2 Perl

Perl provides two libraries for interactions with SOAP

- SOAP::Light
- SOAP::WSDL

2.2.3 IDL

There is no add-on or library to enable SOAP in IDL. Users would have to create ther own SOAP calls.

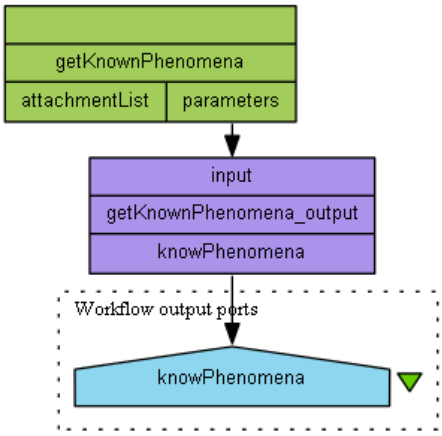
3 Functionality

3.1 getKnownPhenomena

Returns all the phenomena terms the ontology knows about. These strings can be used as input to other functions of the SMS.

WSDL	http://sms.rcs.manchester.ac.uk:8080/helio-sms/sms?wsdl
Operation	getKnownPhenomena
Secure	false
Inputs	
input	none
Outputs	
knowPhenomena	Depth:1 ASCII list of phenomena `getKnownPhenomenaResponse>knowPhenomena`

Example use in workflow:

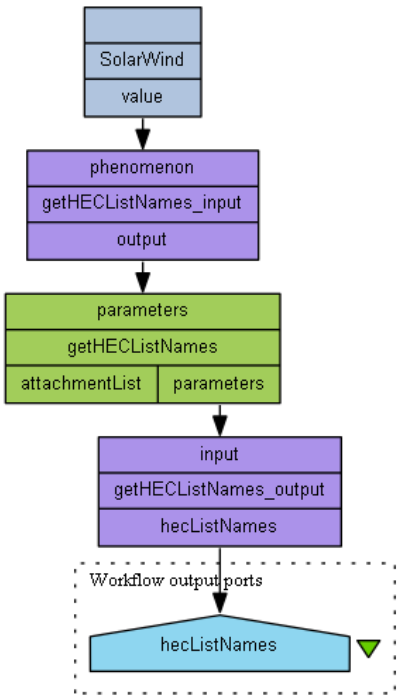


3.2 getHECListNames

Uses the term of a phenomenon to return all HEC catalogues (tables) which contain data about this phenomenon. In case of an unknown term or empty string an empty list will be returned.

WSDL	http://sms.rcs.manchester.ac.uk:8080/helio-sms/sms?wsdl
Operation	getHECListNames
Secure	false
Inputs	
phenomenon	Depth:0 Single ASCII term of a known phenomenon (as returned by getKnownPhenomena) `getHECListNames>phenomenon`
Outputs	
hecListNames	Depth:1 ASCII List of HEC table names `getHECListNamesResponse>hecListNames`

Example use in workflow:

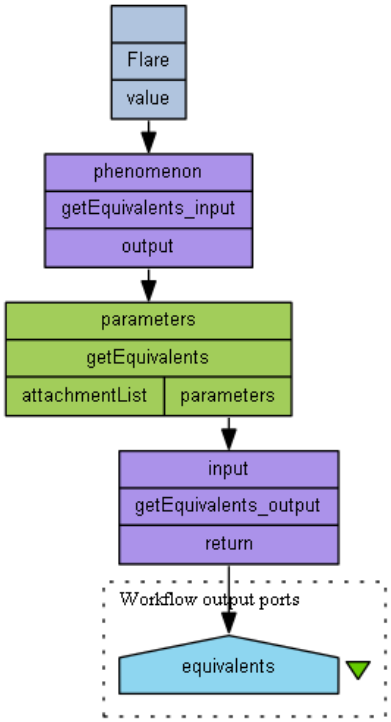


3.3 getEquivalents

This function uses a term from the ontology to query for semantically equivalent terms. A term in the ontology is always equivalent to itself and would be part of the return. If the term is not part of the ontology or an empty string is used as input an empty list would be returned.

WSDL	http://sms.rcs.manchester.ac.uk:8080/helio-sms/sms?wsdl
Operation	getEquivalents
Secure	false
Inputs	
phenomenon	*optional* Depth:0 Single term for which equivalent terms are requested `getEquivalents>phenomenon`
Outputs	
return	*optional* Depth:1 ASCII list of terms in the ontology which are semantically equivalent; empty list in case term does not exist in ontology `getEquivalentsResponse>return`

Example use in Workflow:



3.4 getRelated

This function uses a term from the ontology to query for semantically equivalent terms, child terms, and parent terms. Querying for the term 'Flare' would return: Flare, SolarFlare (equivalent terms), HalphaFlare, SoftXrayFlare, HardXrayFlare (child terms), and Phenomenon (parent term).

WSDL	http://sms.rcs.manchester.ac.uk:8080/helio-sms/sms?wsdl
Operation	getRelated
Secure	false
Inputs	
phenomenon	*optional* Depth:0 Single term for which related terms are requested `getRelated>phenomenon`
Outputs	
return	*optional* Depth:1 ASCII list of terms in the ontology which are either equivalent, child or parent terms to the input; empty list in case term does not exist in ontology `getRelatedResponse>return`

Example use in Workflow: