Agilent GeneSpring Workgroup 5.3 API

User Guide

Agilent Technologies

Notices

Edition

First Edition, October 2006

Printed in USA

Agilent Technologies, Inc.
5301 Stevens Creek Blvd.
Santa Clara, CA 95051

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

In This Guide...

This document contains API (Application Programming Interface) information for data storage extension for Workgroup.

A web service is application or business logic that is accessible using standard Internet protocols such as HTTP. A web service combines the best aspects of component-based development and the World Wide Web. Like components, a web service represents black-box functionality that can be used and reused without regard to how the service is implemented.

The Workgroup SOAP web service can provide access to Workgroup data locally, on an intranet, or remotely, over the internet. The Workgroup web service provides the user with the following functionality:

- Searching for samples
- Retrieving information about samples

This guide contains three chapters:

1 Introducing the Workgroup

In this chapter, you can find information to get started using the Workgroup web service with the Java and Perl programming languages.

2 Workgroup Data Types

This chapter discusses Workgroup Data Types.

3 Workgroup Methods

This chapter contains information about Workgroup web service methods. Provided for each method is the Web Services Description Language (WSDL) definition of the request, and descriptions of the parameter arguments.

Reference

For information on installing Workgroup, and configuration options available at that time, consult the following:

For information on Apache Axis and wsdl2java, visit the Apache Axis website.

For information on SOAP::Lite and SOAP::Data, visit the SOAP::Lite website: http://www.soaplite.com/.

For more information on SOAP and WSDL, consult the http://www.w3.org/ website, the internet or one of the numerous books written on SOAP and WSDL.

Contents

1 Introducing the Workgroup

The Workgroup web service uses Extensible Markup Language (XML)-based technologies such as Simple Object Access Protocol (SOAP) as the communication protocol, and Web Services Description Language (WSDL) as the interface description language. WSDL is an XML-based format for specifying the interface to a web service. The WSDL details the service's available methods and parameter types, as well as the actual SOAP endpoint for the service.

SOAP is the XML-based protocol for sending requests and responses to and from web service methods. It consists of three parts: an envelope defining message contents and processing, encoding rules for application-defined data types, and a convention for representing remote procedure calls and responses.

The Workgroup web service API is implemented to comply with SOAP 1.1 and WSDL 1.1 specifications.

In this chapter, you can find information to get started using the Workgroup web service with the Java and Perl programming languages.

Using Workgroup Methods

To use Workgroup web service methods in your Java code you will need a tool that generates Java classes from the web service's WSDL specification. For example, the Apache Axis jBroker Web wsdl2java compiler transforms WSDL documents into Java RMI (Remote Method Invocation) interfaces. This provides Java programmers with a familiar and typed programming model for invoking and implementing a web service. The client uses a stub to invoke the web service as if it was a regular Java object in the same address space, and the server implements a servlet base class.

If you are programming in Perl, you can use a tool such as SOAP: :Lite, available at http:// www.soaplite.com/. See "Using Workgroup Methods with Perl" on page 10.

Using Workgroup Methods with Java

It is assumed the reader is familar with basic Java programming concepts and already has a Java development environment set up on their on their computer. To get started using Java to invoke the Workgroup web service, take the following steps after Workgroup has been installed on a server you can access:

1. Download a SOAP client library-such as the one that one that comes with your development machine.
2. Convert Workgroup WSDL file to Java classes - convert the Workgroup web service WSDL file to Java classes. The WSDL file can be obtained from Workgroup at: http://<your Workgroup server's URL:Port>/services/wsdl.

Java Data Type Mapping

The table below lists the XML data types supported by the wsdl2java compiler and how they map into Java types. The XML data types are defined in the http://www.w3.org/2001/ XMLSchema namespace. For details about XML data types, refer to the specification at the W3C website. If you use a tool other than wsdl2java, refer to its product information for data mapping specifics.

Table 1 Mapping between XML and Java data types using wsdl2java.

XML Data Type	Java Data Type
anyType	*java.lang.Object*
boolean	*boolean*
byte	*byte*
base64Binary	*byte[]*
hexBinary	*byte[]*
dateTime	*java.util.Calendar*
integer	*java.math.BigInteger*
double	*double*
float	*float*
int	*int*
long	*long*
decimal	*java.math.BigDecimal*
short	*short*
string	*java.lang.String*

Using Workgroup Methods with Perl

One way to implement a Perl SOAP client is to use SOAP::Lite-available at http:// www.soaplite.com/. SOAP: :Lite is a collection of Perl modules providing a simple and lightweight interface to SOAP both on the client and server side. Like wsdl2java for Java, SOAP: :Lite can produce a library of routines directly from a WSDL file. You can use SOAP: :Lite to construct Workgroup web service objects and implement the objects' methods and properties. The following is a simple example.

```
#!perl
use SOAP::Lite;
my $service = SOAP: :Lite
   -->service('http://192.168.123.145/services/Workgroup?wsd
   l') -->proxy('http://192.168.123.151/');
print "Workgroup Version: ", $service->getVersion(), "\n";
```

NOTE If you try to duplicate these Perl examples, remember to subsitute your Workgroup IP address, username and password for the ones used here.

Workgroup URL

You always need to know your Workgroup's IP address, which is configured at installation time. It is not likely to change often. If you are not sure what IP address Workgroup uses, then refer to the deployment guide.

2 Workgroup Data Types

This chapter discusses Workgroup Data Types.

Primitive and Trivial Data Types

The Workgroup SOAP API uses the following primitive XML data types:

Table 2 Primitive XML Data Types used in the GeneSpring SOAP API
Value Description

Primitive XML Data Type	Description
xsd:boolean	Boolean (True / False) values
xsd:date	Date values
xsd:dateTime	Date/time values (timestamps)
xsd:double	Double values
xsd:int	Integer values

These primitive types are the building blocks for the Workgroup data types used in making data storage API calls. To write your client application, you follow the data typing rules defined for your programming language and development environment. The development tool handles the mapping of typed data in your programming language with these SOAP data types.

The Workgroup data types are defined in the Workgroup WSDL file. This chapter provides each types WSDL type definition and a description for each element.

In addition to the types listed in the following table, there are trivial data types that simply represent an array of another data type. These trivial types have names of the form “ArrayOf<primitiveType>”

Table 3 GeneSpring SOAP API Data Types

GeneSpring SOAP API Data Types	Description
Attribute	Contains information about a sample attribute or experiment parameter.
AttributeSearchOptions	Used to define search criteria for the findSamplesByAttribute service. See "findSamplesByAttribute" on page 25 for more information.
SampleInformation	Contains information about a sample's content.
SearchOptions	Used for passing options to various methods that perform searches. Extended by *AttributeSearchOptions*.

Data Types

DataTypeAttribute

This data structure contains information about all attributes stored in the data storage extension.

XML

```
<complexType name="DataTypeAttribute">
    <sequence>
        <element name="attributes" type="string" nillable="true" minOccurs="0"
          maxOccurs="unbounded"/>
        <element name="dataType" type="string" nillable="true"/>
    </sequence>
</complexType>
```

Elements

name: a string containing the attribute name.

dataType : a string that indicates the data and format type for which this attribute belongs.

AttributeSearchOptions

This data structure is used to define search criteria for the findSamplesByAttribute method. This data type extends SearchOptions and inherits its fields.

XML

```
<complexType name="AttributeSearchOptions"> <complexContent>
  <extension base="impl : SearchOptions">
     <sequence>
        <element name="matchAll" type="xsd:boolean"/>
        <element name="searchTerms" nillable="true" type="impl:ArrayOfProperty" />
     </sequence>
  </extension>
```

```
</complexContent>
</ complexType>
```

Elements

matchAll: a boolean—if true, a sample is returned as a match if it meets every one of the search criteria. If false, a sample is returned as a match if it meets any of the search terms.

searchTerms: a Property array containing sample attributes to search against. For each Property in the array, the name field specifies the attribute to search, and the value field specifies the required attribute value.

SampleData

This data structure contains information about a file associated with a data object, such as a sample. The getSampleData method can be used to get the contents of a data file.

XML

```
<complexType name="SampleData">
    <sequence>
        <element name="data" type="base64Binary" nillable="true"/>
        <element name="size" type="long"/>
    </sequence>
</complexType>
```

Elements

Data : the contents of the sample as an attachmen.

Size : the size of the returned sample data.

SampleInformation

This data structure contains information about the contents of a sample. Both getSampleInformation and getMasterAttributesList methods use this data structure.

XML

```
<complexType name="SampleInformation">
    <sequence>
        <element name="attributes" type="tns:Property" nillable="true" minOccurs="0"
           maxOccurs="unbounded"/>
        <element name="dataType" type="string" nillable="true"/>
        <element name="description" type="string" nillable="true"/>
    </sequence>
</complexType>
```

Elements

attributes An array of Attribute objects describing the attributes defined for this sample.

dataType A description of the format type of this sample's raw data files.

description A description this sample's raw data files.

SearchOptions

This data structure is used to pass options to methods that perform searches. It contains flags that affect how the search is done. Another data type, AttributeSearchOptions extends this type and adds further options for particular types of searches.

XML

```
<complexType name="SearchOptions">
  <sequence>
     <element name="caseSensitive" type="xsd:boolean"/> <element name="useWildcards"
        type="xsd:boolean"/>
     <element name="wholeWordsOnly" type="xsd:boolean"/>
  </sequence>
</ complexType>
```

Elements

caseSensitive A boolean specifying whether text matching should be case sensitive

useWildcards A boolean—if true, an asterisk (*) in any search string is treated as a wild-card that matches an arbitrary list of characters.

wholeWordsOnly A boolean—if true, any search string matches only complete words, not pieces of words.

3 Workgroup Methods

This chapter contains information about Workgroup web service methods. Provided for each method is the Web Services Description Language (WSDL) definition of the request, and descriptions of the parameter arguments.

Methods - Summary

The following table summarizes all of the web service methods that are available.

Table 4 Web Service Methods

Method	Description
findSamplesByAttribute	Performs a search for samples based on the values of particular attributes.
getMasterAttributeList	Retrieves the list of attributes from the data store.
getSampleData	Returns the contents of a data file associated with samples..
getSampleInformation	Returns general information about sample objects stored on Workgroup.

Sample Code - Perl Application

Following is a sample of a Perl application.

```
#!perl
use SOAP::Lite;
use Data::Dumper;
use strict;



#
# define constants
#
$SOAP::Constants::DO_NOT_USE_CHARSET = 1;
our $C_URI        = 'http://axis.api.ext.agilent.com/types/';
our $C_PROXY = 'http://localhost:8080/axis2/services/DataService';
our $C_NAMESPACE  = 'http://axis.api.ext.agilent.com/types/';

# ---------------------------
# Main
# ---------------------------
MAIN: {

################################################################################
#                            get_sampleInformation
################################################################################
#
   get_sampleInformation('S_5554');

################################################################################
#                            get_MasterAttributeList
################################################################################
#
   get_MasterAttributeList();

################################################################################
#                            get_sampleData
################################################################################
#
    get_sampleData('PB103894');


################################################################################
#                            find_sampleByAttribute
################################################################################
#
#
my $search_xml = qq(
    <attributeOptions>
      <caseSensitive>true</caseSensitive>
      <wholeWordsOnly>true</wholeWordsOnly>
      <useWildcards>true</useWildcards>
```

```
        <matchAll>true</matchAll>
        <dataType>genotyping</dataType>
        <searchTerms>
           <name>n1</name>
           <value>v1</value>
        </searchTerms>
        <searchTerms>
           <name>n2</name>
           <value>v2</value>
        </searchTerms>
      </attributeOptions>
      );
      my $searchOptions  = SOAP::Data->type('xml' => $search_xml);

      find_sampleByAttribute($searchOptions);

}

#--------------------------------------------------------------------------
# find_sampleByAttribute
#
sub find_sampleByAttribute() {

    my ($searchOptions) = @_;
    my @parameter = ($searchOptions);

    my $som = call_soap('findSamplesByAttribute',  @parameter);

    my @results =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/result');
    my ($results, $result, $i);
    $i = 1;
    foreach $result (@results) {
         my $sampleInfo =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/['.$i.']');
         my @attributes =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/['.$i.']/attribu
tes');
         print_sampleInfo($sampleInfo, @attributes);
         $i++;
    }

}


#--------------------------------------------------------------------------
# get_sampleInformation
#
sub get_sampleInformation() {
   my ($sampleId) = @_;
    my @parameter = (SOAP::Data->name(sampleId =>$sampleId));

    my $som = call_soap('getSampleInformation',  @parameter);

    if ($som->fault) {
```

```
            print  $som->faultstring, "\n";
    } else {
       my $sampleInfo =
$som->valueof('//Body/getSampleInformationResponseElement/result');
       my @attributes =
$som->valueof('//Body/getSampleInformationResponseElement/result/attributes');
       print_sampleInfo($sampleInfo, @attributes);
    }

}

#--------------------------------------------------------------------
# print_sampleInfo
#
sub print_sampleInfo() {

   my ($sampleInfo,@attributes) = @_;

    printf "Sample Id:   %s \n", $sampleInfo->{name}, "\n";
    printf "Data Type:   %s\n", $sampleInfo->{dataType},"\n";
    printf "Description: %s\n\n", $sampleInfo->{description},"\n";

    printf "Attributes: \n";
    my $a;
    foreach $a (@attributes) {
             printf "%-20s = %s \n", $a->{name},$a->{value};
    }
    printf "\n\n";
}

#--------------------------------------------------------------------
# get_MasterAttributeList
#
sub get_MasterAttributeList() {
    my    @parameter;
    my $som = call_soap('getMasterAttributeList',  @parameter);



    if ($som->fault) {
            print  $som->faultstring, "\n";
    } else {

    my @dataTypes =
$som->valueof('//Body/getMasterAttributeListResponseElement/result');
    my ($d, $i);
    $i = 1;
    foreach $d (@dataTypes) {
         printf "Data Type: %s \n----------------------\n", $d->{dataType};
         my @attributes =
$som->valueof('//Body/getMasterAttributeListResponseElement/['.$i.']/attributes');
         $i++;
        foreach $a (@attributes) {
                 printf "%s\n", $a;
        }
```

```
          printf "\n\n";

     }

    }
}

#--------------------------------------------------------------------
# get_sampleData
#
sub get_sampleData() {
   my ($sampleId) = @_;
    my @parameter = (SOAP::Data->name(sampleId =>$sampleId));

    my $som = call_soap('getSampleData',  @parameter);

    if ($som->fault) {
            print  $som->faultstring, "\n";
    } else {
        my $fileName   = $sampleId.'.txt';
       my $sampleData = $som->valueof('//Body/getSampleDataResponseElement/result');

        printf "Sample Data will be saved in File %s \n",$fileName;
        open(OUT, ">".$fileName) or die "Can't open file \n";
        syswrite(OUT,$sampleData->{data}->{Include});
        close(OUT);
   }

}

#--------------------------------------------------------------------
# call_soap
#
sub call_soap() {
     my ($api, @parameter) = @_;

     my $method = SOAP::Data->name($api.'Element')->attr({xmlns => $C_NAMESPACE});
     my $soap = SOAP::Lite
        -> uri($C_URI)
        -> on_action(sub {join '', $api})
#      -> on_debug(sub{print@_})
        -> proxy($C_PROXY);

     my $som = $soap->call( $method => @parameter);
     return $som
}

exit;
```

Methods

Following are the details about the web service methods.

findSamplesByAttribute

This method performs a search for samples based on the values of particular attributes. An AttributeSearchOptions object defines the search criteria.

WSDL

```
<complexType name="findSamplesByAttribute">
    <sequence>
        <element name="attributeOptions" type="tns:AttributeSearchOptions"
nillable="true"/>
    </sequence>
</complexType>
<complexType name="findSamplesByAttributeResponse">
    <sequence>
        <element name="result" type="tns:SampleInformation" nillable="true"
minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Parameters

attributeOptions An AttributeSearchOptions type that defines the sample search criteria.

Return

SampleInformation A SampleInformation type that contains information on all samples matching the search criteria.

Example

```
my $search_xml = qq(
    <attributeOptions>
      <caseSensitive>true</caseSensitive>
      <wholeWordsOnly>true</wholeWordsOnly>
      <useWildcards>true</useWildcards>
      <matchAll>true</matchAll>
```

```
        <dataType>genotyping</dataType>
        <searchTerms>
           <name>n1</name>
           <value>v1</value>
        </searchTerms>
        <searchTerms>
           <name>n2</name>
           <value>v2</value>
        </searchTerms>
      </attributeOptions>
      );
      my $searchOptions  = SOAP::Data->type('xml' => $search_xml);

find_sampleByAttribute($searchOptions);


#--------------------------------------------------------------------------
# find_sampleByAttribute
#
sub find_sampleByAttribute() {

    my ($searchOptions) = @_;
    my @parameter = ($searchOptions);

    my $som = call_soap('findSamplesByAttribute',  @parameter);

    my @results =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/result');
    my ($results, $result, $i);
    $i = 1;
    foreach $result (@results) {
         my $sampleInfo =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/['.$i.']');
         my @attributes =
$som->valueof('/Envelope/Body/findSamplesByAttributeResponseElement/['.$i.']/attribu
tes');
         print_sampleInfo($sampleInfo, @attributes);
         $i++;
    }

}
```

getMasterAttributeList

This method retrieves the master list of attributes that are stored in the Workgroup server storage.

WSDL

```
<complexType name="getMasterAttributeList">
    <sequence/>
</complexType>
<complexType name="getMasterAttributeListResponse">
    <sequence>
        <element name="result" type="tns:DataTypeAttribute" nillable="true"
minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Parameters

None

Return

AttributeList A DataTypeAttribute type that contains information on all the attributes in the data storage extension.

Example

```
get_MasterAttributeList();


#---------------------------------------------------------------------
# get_MasterAttributeList
#
sub get_MasterAttributeList() {
    my    @parameter;
    my $som = call_soap('getMasterAttributeList',  @parameter);



    if ($som->fault) {
            print  $som->faultstring, "\n";
    } else {

    my @dataTypes =
$som->valueof('//Body/getMasterAttributeListResponseElement/result');
    my ($d, $i);
    $i = 1;
    foreach $d (@dataTypes) {
```

```
         printf "Data Type: %s \n--------------------\n", $d->{dataType};
         my @attributes =
$som->valueof('//Body/getMasterAttributeListResponseElement/['.$i.']/attributes');
         $i++;
        foreach $a (@attributes) {
                printf "%s\n", $a;
        }
         printf "\n\n";

    }

   }
}
```

getSampleData

This method returns the raw sample data associated with a sample.

WSDL

```
<complexType name="getSampleData">
    <sequence>
        <element name="sampleIds" type="string" nillable="true"/>
    </sequence>
</complexType>
<complexType name="getSampleDataResponse">
    <sequence>
        <element name="result" type="tns:SampleData" nillable="true"/>
    </sequence>
</complexType>
```

Parameters

sampleIds An array of sample IDs that defines the sample search criteria.

Return

SampleData A SampleData type that contains information about the raw data object corresponding to the sample.

Example

```
#--------------------------------------------------------------------
# get_sampleData
#
```

```
sub get_sampleData() {
   my ($sampleId) = @_;
    my @parameter = (SOAP::Data->name(sampleId =>$sampleId));

    my $som = call_soap('getSampleData',  @parameter);

    if ($som->fault) {
            print  $som->faultstring, "\n";
    } else {
        my $fileName   = $sampleId.'.txt';
       my $sampleData = $som->valueof('//Body/getSampleDataResponseElement/result');

        printf "Sample Data will be saved in File %s \n",$fileName;
        open(OUT, ">".$fileName) or die "Can't open file \n";
        syswrite(OUT,$sampleData->{data}->{Include});
        close(OUT);
   }

}
```

getSampleInformation

This method returns the general Sample information about the sample object that is stored in the workgroup database.

WSDL

```
<complexType name="getSampleInformation">
    <sequence>
        <element name="sampleIds" type="string" nillable="true"/>
    </sequence>
</complexType>
<complexType name="getSampleInformationResponse">
    <sequence>
        <element name="result" type="tns:SampleInformation" nillable="true"/>
    </sequence>
</complexType>
```

Parameters

sampleIds An array of sample ids that defines the sample search criteria.

Return

SampleInformation A SampleInformation type that contains information on all samples matching the search criteria.

Example

```
get_sampleInformation('S_5554');


#--------------------------------------------------------------------------
# get_sampleInformation
#
sub get_sampleInformation() {
   my ($sampleId) = @_;
    my @parameter = (SOAP::Data->name(sampleId =>$sampleId));

    my $som = call_soap('getSampleInformation',  @parameter);

    if ($som->fault) {
            print  $som->faultstring, "\n";
    } else {
       my $sampleInfo =
$som->valueof('//Body/getSampleInformationResponseElement/result');
       my @attributes =
$som->valueof('//Body/getSampleInformationResponseElement/result/attributes');
       print_sampleInfo($sampleInfo, @attributes);
    }

}
```

Index

Index

www.agilent.com

In This Book

API (Application Programming Interface) information for data storage extension for Workgroup.

Information about Workgroup web service methods and data types.

First edition, October 2006

Agilent Technologies