# Natural Insight Web Services User Guide

*Published on Tuesday, May 05, 2015 at 2:07 PM*

# Table of Contents

Chapter 1

# Natural Insight Web Services

# Natural Insight Web Services Introduction

Companies today need secure and reliable means to tie their myriad systems and data together. How do you integrate your Natural Insight operations and data with your company's existing business processes and infrastructure? Leverage Natural Insight Web Services.

Natural Insight Web Services enable data integration with internal and third-party applications through remote access. Our services provide APIs – open-standard methods for system-to-system communication between Natural Insight and other products including:

- HR and payroll platforms (Kronos, ADP, Paychex, etc.),
- accounting and billing software (Quickbooks, etc.),
- supply chain distribution systems,
- order management and inventory applications,
- ERP software, and
- CRM systems (Salesforce.com, etc.).

Clients can exploit Natural Insight Web Services APIs to quickly develop new mashups with other web services or to extend your current applications for expansion and adaptation with NI-powered workforce scheduling, management and reporting.

## Developer's Guide to Natural Insight Web Services

If you are a company with your own dedicated team of internal application developers, this guide will walk your development team through Natural Insight's current documented Web Services. The offerings include NI resource management services (Import) and NI data access (Export) APIs.

### Natural Insight Import Web Services

- Location Maintenance (create and update locations API)
- Staff Maintenance (create and update staff members API, which also includes two export methods for confirmation and reporting of staff member data operations*)

- Visit Creation (create visits API)
- Single Sign On (SSO) – Remote Access

### Natural Insight Export Web Services

- NI Packager (list API)
- Deleted Visits (list API)
- *Staff Maintenance contains two methods for exporting Natural Insight staff member data - *getStaff* and *getStaffMaintReport*.

## Partnership with Natural Insight Client Services

Your dedicated Natural Insight Client Services account manager is your partner in analyzing your business objectives and recommending integration strategies between Natural Insight and your current IT infrastructure. If you are a company who does not have a dedicated team of developers, Natural Insight Client Services can work with you to define and develop your custom data integration goals.

Natural Insight Client Services can recommend custom development through the Natural Insight Technical Services team or can consult with third-party developers. The cycle of a custom integration project includes:

- an audit of current system configuration,
- a needs assessment,
- a project definition including budget with client sign-off,
- development,
- quality assurance testing,
- implementation, plus
- additional services such as training and documentation on an as-needed basis.

Whatever your requirements, we are here to help. Consult with your dedicated Natural Insight Client Services account manager today to discuss how we can assist you with your system integration needs.

# Natural Insight Web Services Overview

The purpose of this technical guide describing Natural Insight Web Services is to detail to developers the methods available for integration of Natural Insight data with third-party applications. The audience for this document is your company's internal technical development team, or, alternatively, the third-party development team with whom you have partnered to facilitate integration of your IT systems with Natural Insight.

Natural Insight provides SOAP Web APIs described in WSDL for the different web service integrations with Natural Insight. Before proceeding, let's define the basic components of Natural Insight Web Services.

## What is a Web API?

A Web API (Application Programming Interface) is a series of programming instructions and standards for communicating with a Web-based software application released to a service requestor so that software developers can more easily access the service provider's proprietary data and design products powered by the service provider's services and data.

### Who is the Service Provider?

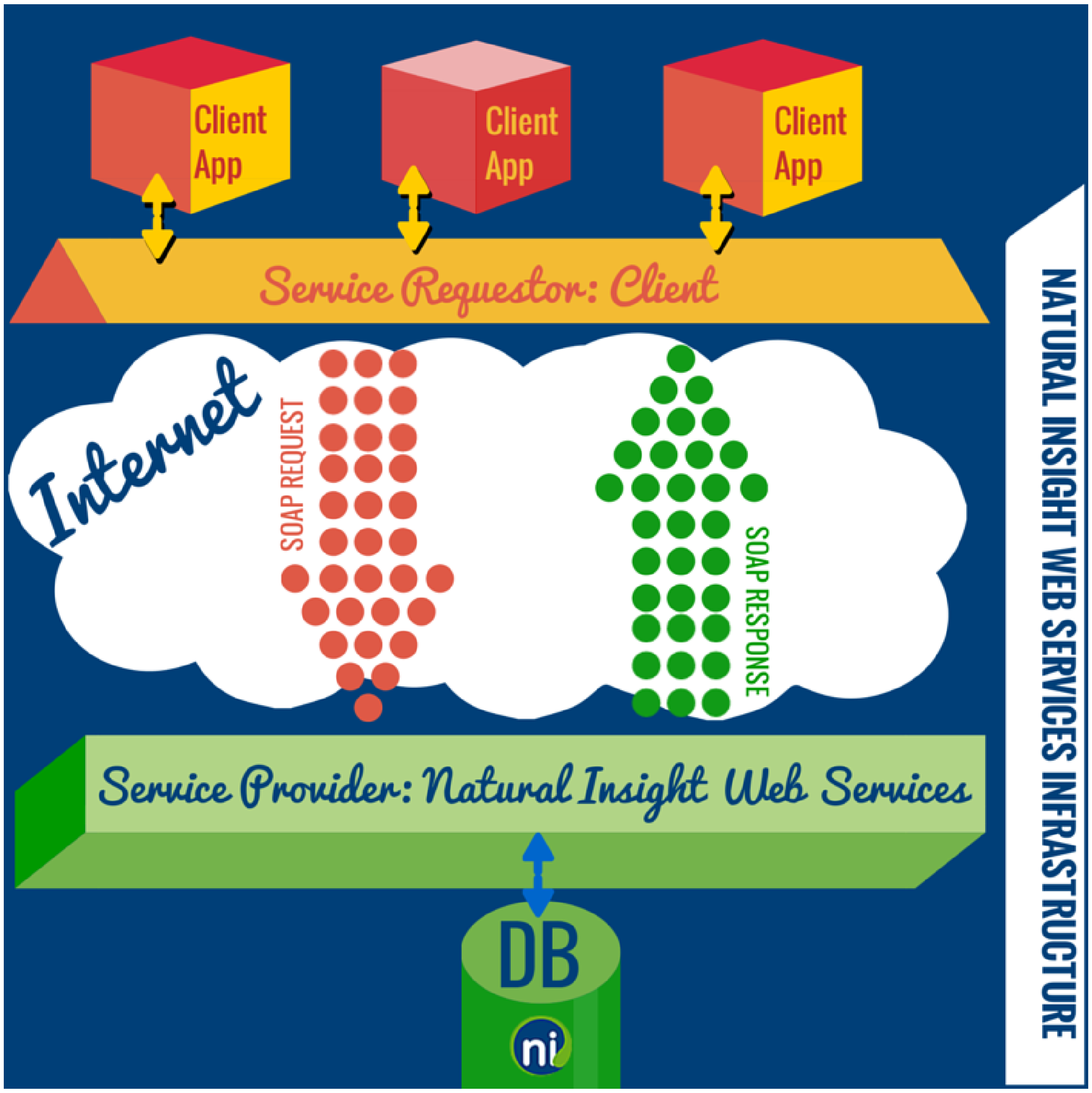
Natural Insight is the service provider in the Natural Insight Web Services model.

### Who is the Service Requestor?

Any Natural Insight client can be a service requestor.

## What is SOAP?

SOAP stands for Simple Object Access Protocol and is a standard W3C-recommended XML messaging protocol for Web Services to swap structured data over the Internet. The service requestor - any NI client - can issue a SOAP request to the service provider (Natural Insight) and, in reply, the service provider (Natural Insight) sends a SOAP response to the service requestor (the NI client who made the request).


Client App
Client App
Client App
Service Requestor: Client
Internet
SOAP REQUEST
SOAP RESPONSE
Service Provider: Natural Insight Web Services
DB
ni
NATURAL INSIGHT WEB SERVICES INFRASTRUCTURE

SOAP is the open standards protocol for encoding XML messages used to transport data to any OS (operating system) over any type of network protocol.

### What is XML?

XML stands for eXtensible Markup Language and is a language used to code (represent) and decode data transmitted across systems.

## What is WSDL?

In a SOAP-based Web Service, the service is described in WSDL documents. WSDL stands for Web Service Description Language and is a W3C-recommended XML-based language which simply defines all functions and structures of the Web Service at hand, including locating and describing web services and how to access them.

## Prerequisites for Natural Insight Web Services

In order to access Natural Insight Web Services, the following prerequisites must be met:

1. You must be an active Natural Insight client.

2. You must have a activated Natural Insight authorization code - authCd - issued by Natural Insight Client Services. If you do not have a current authorization code, contact Natural Insight Client Services to obtain one. This authorization code is a unique key that identifies each Natural Insight client and is used for all Natural Insight Web Services. As the value of this authorization code cannot easily be changed (since it would affect all Web Service integration for the client), never expose this code to Natural Insight end users.

3. Though not a requirement, Natural Insight recommends accessing Natural Insight Web Services over SSL.

# Natural Insight Import Web Services

Clients can automate several import processes within Natural Insight using the following Natural Insight Web Services:

- Location Maintenance (create and update locations APIs)
- Staff Maintenance (create and update staff members APIs)
- Visit Creation (create visits API)
- Single Sign On (SSO) – Remote Access

## Location Maintenance

The Location Maintenance import service provides a secure means for external IT systems to push data about locations where work is performed into Natural Insight. Using unique and secret identifiers, you can add and update information about locations in your client instance of Natural Insight. You can create unique Location Maintenance templates based upon your business needs.

## Staff Maintenance

The Staff Maintenance import service provides a secure means for external IT systems to push data about the staff members who perform work into Natural Insight. Using unique and secure identifiers, you can add and update information about staff members in your client instance of Natural Insight.

## Visit Creation

The Visit Creation[1] import service provides a secure means for external IT systems to create visits (or tasks, calls, etc. depending on your company's vocabulary) into Natural Insight. Using unique and secure identifiers, you can add visits (or tasks, calls, etc.) in your client instance of Natural Insight.

## Single Sign On (SSO) Remote Access

The Single Sign On service provides a secure means for external IT systems to offer a remote single sign on to Natural Insight. Using unique and secure identifiers, you can create sign in pages that allow your company staff to directly log in to Natural Insight without having to use the Natural Insight log in page.

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

# Natural Insight Export Web Services

Natural Insight allows clients to export a variety of information out of their instance of the product. Clients can automate the export process from Natural Insight using Natural Insight Packager, which also includes a separate component for Deleted Visits. In addition, the *getStaff* and *getStaffMaintReport* methods of the Staff Maintenance web service export staff data in your instance of Natural Insight.

## Natural Insight Packager

The Natural Insight Packager web service exports XML data at the visit[1] level according to three possible search options:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

## Packager Deleted Visits

The Natural Insight Packager Deleted Visits web service returns a package of deleted visits falling within a requested date range.

## getStaff and getStaffMaintReport Methods of the Staff Maintenance Web Service

The Staff Maintenance web service includes an import service via the *receiveData* method as well as an export service via the *getStaff* and *getStaffMaintReport* methods that export information for a single staff member at a time (*getStaff*) and export a report on staff maintenance operations during a specified date range (*getStaffMaintReport*).

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

# Chapter 2

# IMPORT WEB SERVICES

Chapter 2

# Location Maintenance Web Service

# Location Maintenance Web Service Overview

The Location Maintenance Web Service provides clients a secure means of adding and updating locations through an API. Each call to the Location Maintenance Web Service will either add or update one location record at a time (unlike the bulk location maintenance process via spreadsheet where several location records can be added and/or updated simultaneously).

## Description

Add a location record or update a location's information in real-time remotely using XML provided by Natural Insight. This import process can be performed at any time and can be triggered by a target third-party system if needed.

### Frequency

On-demand

## WSDL

You can access the WSDL for the Location Maintenance API at https://my.naturalinsight.com/maintenanceWebService.cfc?wsdl . The Location Maintenance WSDL code is as follows:

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://DefaultNamespace" xmlns:intf="http://DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://DefaultNamespace">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">

<import namespace="http://xml.apache.org/xml-soap"/>

<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType name="CFCInvocationException">
```

```
<sequence/>
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="recieveDataRequest">
<wsdl:part name="templateId" type="xsd:double"/>
<wsdl:part name="templateToken" type="xsd:string"/>
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="data" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="recieveDataResponse">
<wsdl:part name="recieveDataReturn" type="apachesoap:Document"/>
</wsdl:message>
<wsdl:portType name="maintenanceWebService">
<wsdl:operation name="recieveData" parameterOrder="templateId templateToken
authCd data">
<wsdl:input message="impl:recieveDataRequest" name="recieveDataRequest"/>
<wsdl:output message="impl:recieveDataResponse" name="recieveDataResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="maintenanceWebService.cfcSoapBinding"
type="impl:maintenanceWebService">
<wsdlsoap:binding style="rpc" trans-
port="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="recieveData">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="recieveDataRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="recieveDataResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:output>
```

```
<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://DefaultNamespace"
use="encoded"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="maintenanceWebServiceService">

<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Generic
Templated import Web Service Component</wsdl:documentation>

<wsdl:port binding="impl:maintenanceWebService.cfcSoapBinding"
name="maintenanceWebService.cfc">

<wsdlsoap:address
location="https://staging.naturalinsight.com/maintenanceWebService.cfc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

# Location Maintenance Data Information

Since locations can have a large number of diverse fields that characterize them, Natural Insight separates the fields into three categories: required, optional and custom (user-created).

## Required Location XML Attributes

Certain <location> XML attributes are required when performing add or update location database operations.

### Required Location XML Attributes When Adding a New Location to Natural Insight

The two <location> XML attributes required when adding a new location to Natural Insight are the following:

1. location ID (*locCode*)
2. location name (*name*)

See **Location Maintenance Required and Optional Data Descriptions** on page 26 for more information about each location XML attribute.

### Required Location XML Attribute When Updating a Location in Natural Insight

The one <location> XML attribute required when updating a location in Natural Insight is the following:

- location ID (*locCode*)

See **Location Maintenance Required and Optional Data Descriptions** on page 26 for more information about each location XML attribute.

## Optional Location XML Attributes When Adding or Updating Locations in Natural Insight

The optional <location> XML attributes available when adding a new location or updating a current location to Natural Insight are the following:

1. location number (*locNumber*)
2. location street address (*street*)
3. location city (*city*)
4. location state or province (*stateProvince*)
5. location postal code (*postalCode*)
6. location country (*countryCode*)
7. location phone (*phone*)
8. location default manager (*mgr_1*)*
9. location default staff member (*dflt_1*)**
10. location average hourly labor cost (*averageHourlyLaborCost*)
11. location chain (*chain*)
12. location description (*description*)
13. location email (*email*)
14. location extra effort amount (*extraEffortAmount*)
15. location extra effort type (*extraEffortType*)
16. location fax (*fax*)
17. location group code (*groupCode*)
18. location source (*source*)
19. location latitude (*latitude*)
20. location longitude (*longitude*)

21. location phone readback (*phoneReadback*)

22. location status (*status*)

23. new location ID (*newLocCode*) - This is used if you need to update the current location ID (*locCode*) to a new location ID.

See **Location Maintenance Required and Optional Data Descriptions** on page 26 for more information about each location XML attribute.

## Custom Location Items Created by Your Company's Natural Insight Administrator

In addition to optional <location> XML attributes, each Natural Insight client has the ability to create the following custom items (which are also optional) to describe and categorize each location - tags, attributes, manager roles and default staff roles. If these optional <location> items don't apply to your company's locations, there is no need to create them.

### Tags

- You can create any custom tag to describe or provide information about each location. Location tags are used to define information options about locations that are limited to one or more of several available prescribed values. A location tag can either have only one single value associated with the tag ("either or" behavior), or, alternatively, multiple values associated with the tag ("choose all that apply" behavior).

- Since tags are optional, they can be left blank - i.e. their values don't have to be defined - for individual locations.

- One example of a location tag is <environment> with possible prescribed values being "indoor mall store" / "outdoor mall store" / "indoor mall kiosk" / "outdoor mall kiosk" / "airport mall kiosk" / "other". The <environment> tag can be defined or not defined for a particular location. If the <environment> tag is defined for a particular location, you can specify that the tag have only one corresponding value, or, alternatively, may accept multiple values. In the latter scenario, you may have a store that has two main entrances - one in the interior of a mall ("indoor mall store") and one on the exterior of a mall ("outdoor mall store"). As a result, if the <environment> tag is setup to accept multiple values, the tag could include the two values of "indoor mall store" and "outdoor mall store".

- The process of creating (and editing) tags is completed via **LOCATIONS > Tags**.

- No default location tag exists within Natural Insight. Any tags must be created by an NI administrator within your company.
- Location tags can be used to search for locations in the Location Search interface. For example, an NI administrator within your company could perform a search of all locations possessing the “airport mall kiosk” tag and create visits/tasks at only these locations for a special “Travel to NYC” promotion.

## Attributes

- You can create any custom attribute to describe or provide information about each location. Location attributes are used to record free-form information about locations that do not have to adhere to one or more options from a collection of possible values.
- Since attributes are optional, they can be left blank - i.e. their values don’t have to be defined - for individual locations.
- One example of a location attribute is <notes>. Since the value of a location’s <notes> attribute could be any string of text under the sun - such as “Under renovation from January 15 - March 15, 2015 though still open for business” or “Must sign-in before beginning any work on site” for example - location notes should be recorded as values of an attribute rather than a tag for each location.
- The process of creating (and editing) attributes is completed via **ADMINISTRATION > Location Attributes**.
- No default location attribute exists within Natural Insight. Any attributes must be created by an NI administrator within your company.
- Location attributes cannot be used to search for locations in the Location Search interface.

***DIFFERENCE BETWEEN LOCATION TAGS AND LOCATION ATTRIBUTES***

The two main differences between location tags and location attributes is:

1. Location attributes can accept values with free-form information such as any integer, any date or any string of text, for example, whereas location tags can accept only one or more options of a collection of acceptable values.
2. Location tags can be used to search for locations in the LOCATIONS > Location Search interface while location attributes cannot. You can search for locations that match

specific location tag values (for example, search for all locations with an <environment> tag value of “indoor mall kiosk”). It is not possible to search for locations using location attributes via the Location Search interface.

## *Manager Roles

- Different manager roles can be created for each location.
- By default, the Default Manager role (location *mgr_1*) exists as an element for each location in Natural Insight. Since location *mgr_1* is an optional element, its value can be left blank for any location.
- In addition to the Default Manager role, you may add additional custom default manager roles - such as Default Pharmacy Manager for the location, Default Merchandising Manager for the location, Default Reset Manager for the location, etc. - that will be available for each location in your client instance of Natural Insight. This process is completed via **LOCATIONS > Manager Roles**. Any custom default manager roles created through this process will be optional and, thus, their values can be left blank for individual locations.
- Designating different custom manager roles for each location allows you to have different managers in charge of different types of work - i.e. visits/calls/tasks/etc. depending on your company’s lingo - at a location. Each of the managers specified at a location can have separate staff reporting to them. Visits/calls/tasks/etc. at a location can be created for any staff managed by a particular manager role.
  - For example, if a reset project is created for the nationwide reset team to perform resets across US Targets, a Reset Manager role can be created and assigned for each Target store. Then, the reset work - the visits/calls/etc. depending on your company lingo - and the staff scheduled for the reset visits can be overseen by the assigned Reset Manager at each Target location. To define this in Natural Insight, you designate the Manager Role as the Reset Manager when you create visits for all of the Targets nationwide.

## **Default Staff Member Roles

- Different default staff member roles can be created for each location.
- By default, the Default Staff Member role (*dflt_1*) exists as an element for each location in Natural Insight. Since *dflt_1* is an optional element, its value can be left blank for any location.

- The Default Staff Member role (*dflt_1*) at each location can be assigned to an individual staff member. For example, if you have a staff member who always services a location and does a great job, you can appoint this person as the Default Staff Person (*dflt_1*) for that location and choose to offer them opportunities before everyone else.

- In addition to the default staff person role, you may add additional custom default staff person roles - such as Default Pharmacy Staff Person, Default Merchandising Staff Member, Default Reset Staff Person, etc. - that will be available for each location in your client instance of Natural Insight. This process is completed via **LOCATIONS > Default Staff**. Any custom default staff person roles created through this process will be optional and, thus, their values can be left blank for individual locations.

  - For example, if you have Jane Smith who specializes in demonstration and tasting work and you want to send Jane to staff any demonstration or tasting visits at locations near her home, you can do the following:

    1. Create a Default Demo/Tasting Staff Member role in Natural Insight for each location.

    2. Assign Jane Smith to that role at the locations where she is available to work/visit. In this way, whenever you have demonstration and/or tasting visits to staff in her area, you can send these work opportunities to Jane first before offering them to other staff members.

# Location Maintenance Required and Optional Data Descriptions

Access the Location Maintenance XML-based WSDL at https://my.naturalinsight.com/maintenanceWebService.cfc?wsdl.

| Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| n/a | *authCd* | Required | | | Authorization code. Obtain from Natural Insight Client Services. |
| Location ID | *locCode* | Required | | | |
| New Location ID | *newLocCode* | Optional | | | |
| Location Number | *locNumber* | Optional | | | |
| Location Name | *name* | Optional | | | |
| Location Description | *description* | Optional | | | |
| Location Status | *status* | Optional | | | |
| Location Chain | *chain* | Optional | | | |
| Location Street Address | *street* | Optional | Char | 500 | All street address information is to be contained in this field. There is no other street address field. |
| Location City | *city* | Optional | Char | 200 | |
| Location State or Province | *stateProvince* | Optional | Char | 2 | 2-letter code |
| Location Postal Code | *postalCode* | Optional | Char | 15 | 5 or 9 digit zip code |
| Location Country | *countryCode* | | | | |

| Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Location Latitude | *latitude* | Optional | Float | 12 | (if available) Valid home latitude in decimal degrees. Latitude measurements range from 0° to (+/–) 90°. |
| Location Longitude | *longitude* | Optional | Float | 12 | (if available) Valid home longitude in decimal degrees. Longitude measurements range from 0° to (+/–) 180°. |
| Location Phone | *phone* | Optional | Char | 25 | |
| Location Email | *email* | Optional | Char | 250 | |
| Location Fax | *fax* | Optional | | | |
| Location Average Hourly Labor Cost | *averageHourlyLaborCost* | Optional | | | |
| Location Extra Effort Amount | *extraEffortAmount* | Optional | | | |
| Location Extra Effort Type | *extraEffortType* | Optional | | | |
| Location Group Code | *groupCode* | Optional | | | |
| Location Phone Readback | *phoneReadback* | Optional | | | |
| Location Source | *source* | Optional | | | |
| Location Default Manager | *mgr_1* | Optional | | | |
| Location Default Staff Member | *dflt_1* | Optional | | | |

naturalinsight®

# Location Maintenance Custom Templates

Due to the multitude of optional and custom variables that can be created to describe locations within the platform, Natural Insight allows you to build your own templates in order to add new locations or update existing locations. In this way, you must choose which pieces of information (<elements>) you will define for your locations. Which optional fields will you specify for each location? Which custom elements - tags, attributes, manager roles or default staff roles - will you designate for each?

In order to work with the Location Maintenance web service, a Location Maintenance custom template - with its corresponding XML data structure, identifier and security key - is required.

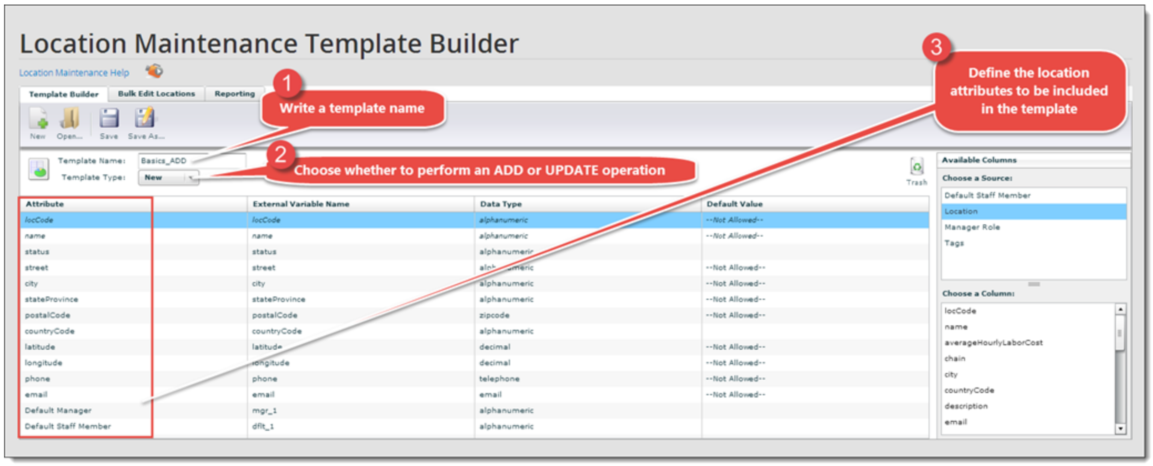
## Custom Location Templates to Add or Update Locations

As a result of this variability, the Location Maintenance Web Service requires that you create a template using the **Location Maintenance Template Builder** interface (**LOCATIONS > Location Template Builder**) in Natural Insight. In this GUI, you will determine:

1. the template name

2. the template type - whether the template will perform an ADD or UPDATE operation for your locations

3. the location attributes to define in this template



Template Name: Basics_ADD
Template Type: New

| Attribute | External Variable Name | Data Type | Default Value |
|---|---|---|---|
| *locCode* | *locCode* | *alphanumeric* | *--Not Allowed--* |
| *name* | *name* | *alphanumeric* | *--Not Allowed--* |
| status | status | alphanumeric | |
| street | street | alphanumeric | --Not Allowed-- |
| city | city | alphanumeric | --Not Allowed-- |
| stateProvince | stateProvince | alphanumeric | --Not Allowed-- |
| postalCode | postalCode | zipcode | --Not Allowed-- |
| countryCode | countryCode | alphanumeric | |
| latitude | latitude | decimal | --Not Allowed-- |
| longitude | longitude | decimal | --Not Allowed-- |
| phone | phone | telephone | --Not Allowed-- |
| email | email | email | --Not Allowed-- |
| Default Manager | mgr_1 | alphanumeric | |
| Default Staff Member | dflt_1 | alphanumeric | |

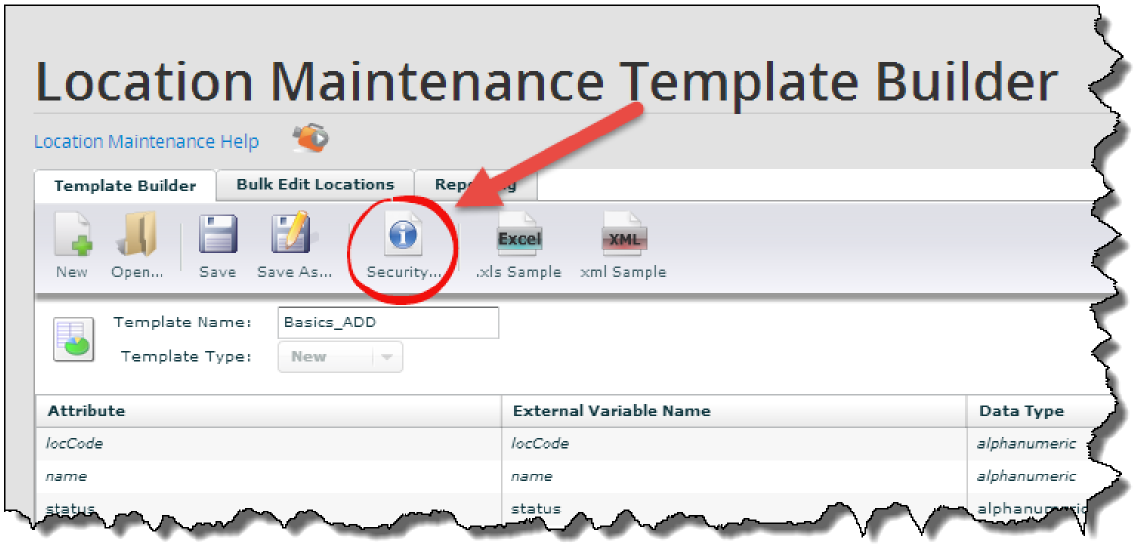
4. the template ID (*templateId*) issued once the template is saved

5. the security code (*templateToken*) issued once the template is saved

6. the XML packet (*data*), which matches the template and its corresponding attributes, issued once the template is saved.

## Creating and Saving a Location Maintenance Template

Once you have performed steps 1-3 detailed above, you will save your template by clicking the **Save** button.



Template Name: Basics_ADD
Template Type: New

| Attribute | External Variable Name | Data Type |
|---|---|---|
| locCode | locCode | alphanumeric |
| name | name | alphanumeric |
| status | status | alphanumeric |

## Obtaining the templateId, templateToken and XML Data of the Location Maintenance Template

In order to perform a Location Maintenance import ADD or UPDATE operation via Natural Insight Web Services, you will need - an authorization code (*authCd*), a template ID (*templateId*), a security code (*templateToken*) and XML data for the location (*data*) - for the *receiveData* method in the Location Maintenance web service.

### *receiveData Method Attribute Descriptions*

The *receiveData* method of the Location Maintenance web service requires four attributes.

1. Authorization code (*authCd*) - a unique key that identifies the Natural Insight client. This *authCd* is used for all Natural Insight Web Services and should never be exposed to users. This value cannot be easily changed as it would affect all Natural Insight Web Services integration for that client.

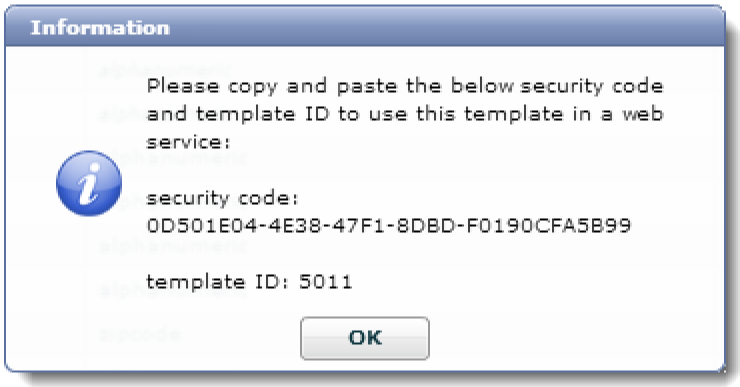
2. Template ID (*templateId*) - a unique integer identifier for a given Location Maintenance template.

3. Security code (*templateToken*) - a second universally unique identifier (UUID) string used to validate access to the template identified by the template ID in #2. It is up to the client to ensure the *templateId* and the *templateToken* match.

4. XML data (*data*) - an XML packet, matching the Location Maintenance template identified by the template ID in #2, for a single location. All XML attributes specified in the template must be passed in though they may be passed in as empty strings. Note that passing in empty strings for an attribute will remove the specific data value from the affected location record for that attribute.

## *How to Obtain the templateId and templateToken*

You can obtain the *templateId* and *templateToken* by clicking on the **Security** icon button on the **Location Maintenance Template** page. Upon clicking the **Security** icon, an **Information** popup box appears with the template ID (*templateId*) and the security code (*templateToken*).



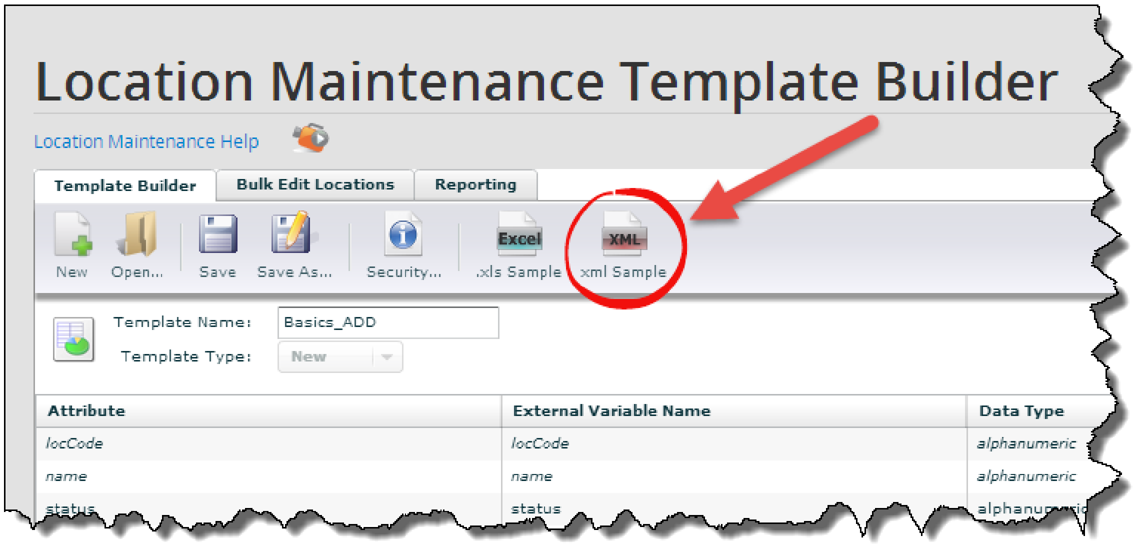
Please copy and paste the below security code and template ID to use this template in a web service:

security code:
0D501E04-4E38-47F1-8DBD-F0190CFA5B99

template ID: 5011

## *How to Obtain the XML Location Maintenance Data*

You can access these two attributes plus the XML data as well by downloading an XML sample of the template you just created. Click the **XML Sample** icon button to download the XML sample file named *Sample_of_XML_format.xml*.



|  |  |
|---|---|
| Template Name: | Basics_ADD |
| Template Type: | New |

| Attribute | External Variable Name | Data Type |
|---|---|---|
| locCode | locCode | alphanumeric |
| name | name | alphanumeric |
| status | status | alphanumeric |

The contents of this file will vary depending on the optional and/or custom variables you have included in your template. An example of the code content of an .xml sample file - with the *templateId* and *templateToken* found in the <root> element - is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<root templateId="5011" templateToken="0D501E04-4E38-47F1-8DBD-F0190CFA5B99">
   <location locCode="" name="" status="" street="" city="" stateProvince=""
   postalCode="" countryCode="" latitude="" longitude="" phone="" email=""
   mgr_1="" dflt_1="" />
</root>
```

Once you have received the *templateId*, *templateToken* and the XML data structure, you can configure these items, along with the authorization code, for the *receiveData* method of the Location Maintenance web service.

# Location Maintenance Web Service receiveData Method

The Location Maintenance web service can be consumed using either the SOAP protocol or simple HTTP GET/POST requests. There is only one public method in the Location Maintenance web service: *receiveData*.

## Process for Calling the receiveData Location Maintenance Method

The general process for calling the *receiveData* process is as follows:

1. An NI Administrator at your company requests from Natural Insight Client Services a unique authorization code (*authCd*) for your client instance of Natural Insight to access and employ Natural Insight Web Services.

2. The NI Administrator, who will have permissions to the **Location Maintenance Template Builder** interface (**LOCATIONS > Location Template Builder**), builds a custom template to obtain this template's template ID (*templateId*), security code (*templateToken*) and XML Location Maintenance attributes.

3. The client programs code to pass a single location's data at a time according to the XML data structure (*data*) received from the **Location Maintenance Template Builder** interface by clicking on the **XML Sample** icon button.

4. In the code, the client system calls the Location Maintenance web service's *receiveData* method passing in the four required parameters:

    1. Authorization code as a string (*authCd*)

    2. Template ID as an integer (*templateId*)

    3. Security code as a string (*templateToken*)

4. XML location packet as a string (data)

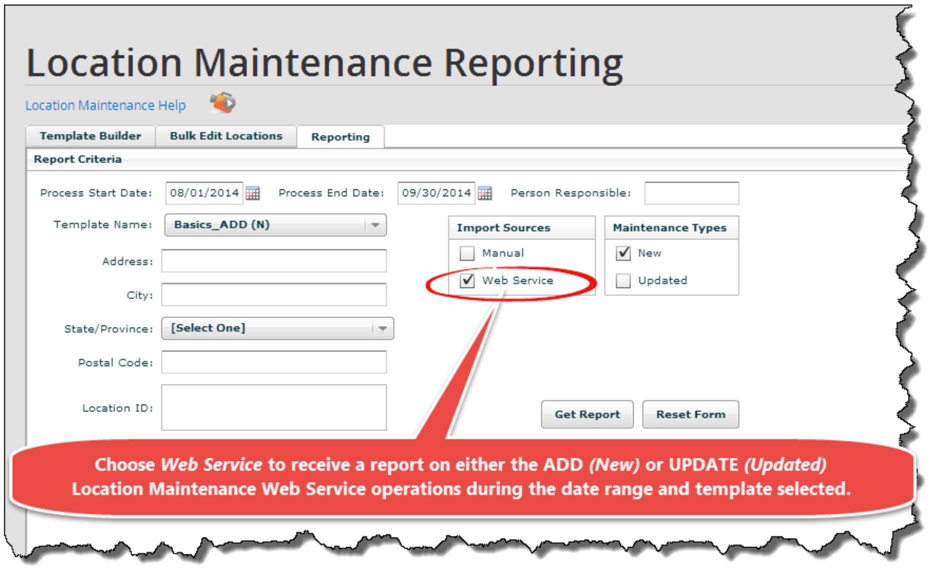
***IMPORTANT NOTES ON XML LOCATION DATA (DATA)***

a. The XML location data (*data*) passed in for each call to the *receiveData* method will process a single import - ADD or UPDATE - location record at a time.

b. The XML location data (*data*) is required - i.e. it must be passed in - but may contain empty string values for the referenced attributes.

5. If the parameters are valid, Natural Insight attempts to add or update location by location using the XML data provided.

6. Whether the operation is successful or not, the results of the remote add or update can be viewed via report by using the Location Maintenance Reporting interface (**LOCATIONS > Location Maintenance Reports**).



Process Start Date: 08/01/2014
Process End Date: 09/30/2014
Person Responsible:
Template Name: Basics_ADD (N)
Address:
City:
State/Province: [Select One]
Postal Code:
Location ID:

Import Sources: Manual [✓ Web Service]
Maintenance Types: [✓ New] Updated

Choose *Web Service* to receive a report on either the ADD *(New)* or UPDATE *(Updated)* Location Maintenance Web Service operations during the date range and template selected.

7. If the ADD or UPDATE operation was not successful, potential error messages include:

- Bad or missing authCd:

```
<return>Upload Failed: Authentication Failure.</return>
```

- Template Not Found:

```
<return>Upload Failed: Template not found.</return>
```

- Data errors: Not returned by the service.

# Location Maintenance Web Service API Summary

## URL

https://my.naturalinsight.com/maintenanceWebService.cfc

## WSDL

https://my.naturalinsight.com/maintenanceWebService.cfc?wsdl

## Method

### receiveData

```
<wsdl:operation name="recieveData" parameterOrder="templateId templateToken
authCd data">
```

#### *receiveData Parameters*

- required numeric *templateId*
- required string *templateToken*
- required string *authCd*
- required string *data*

#### *receiveData Description*

The Location Maintenance *receiveData* method processes a single import - ADD or UPDATE - location record based on the four required supplied parameter values. The attributes required by the template must be passed in but may contain empty string values.

#### *Potential Error Messages*

- Bad or missing authCd:

```
<return>Upload Failed: Authentication Failure.</return>
```

- Template Not Found:

```
<return>Upload Failed: Template not found.</return>
```

- Data errors: Not returned by the service.

Chapter 3

# Staff Maintenance Web Service

# Staff Maintenance Web Service Overview

The Staff Maintenance Web Service provides clients a secure means of adding and updating staff members through an API. Each call to the Staff Maintenance Web Service will either add or update one record at a time (unlike the bulk staff maintenance process via spreadsheet where several staff records can be added and/or updated simultaneously).

Once a staff record is created/added or updated (including termination), the operation can be confirmed on a one-by-one basis with a read function - i.e. the *getStaff* method. In addition, the *getStaffMaintReport* method will report on all staff maintenance operations - adds, updates, terminations - during a specified date range.

## Description

Add a staff member record or update a staff member's information in real-time remotely using XML provided by Natural Insight. This import process can be performed at any time and can be triggered by a target HR (Human Resources) system. Confirm a staff add/update with a read function (*getStaff*) delivering XML staff data per staff person contained in the NI platform. Report on staff maintenance operations - adds, updates, terminations, etc. - with a report function *getStaffMaintReport*.

### Frequency

On-demand

## WSDL

You can access the WSDL for the Staff Maintenance API at https://my.naturalinsight.com/staffMaintWebService.cfc?wsdl.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://niWeb" xmlns:intf="http://niWeb"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://niWeb">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>
```

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<complexType name="CFCInvocationException">
<sequence/>
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="getStaffRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="staffNumber" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="getStaffResponse">
<wsdl:part name="getStaffReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getStaffMaintReportResponse">
<wsdl:part name="getStaffMaintReportReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="receiveDataRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="rowType" type="xsd:string"/>
<wsdl:part name="staffNumber" type="xsd:string"/>
<wsdl:part name="firstName" type="xsd:string"/>
<wsdl:part name="lastName" type="xsd:string"/>
<wsdl:part name="supervisorKey" type="xsd:string"/>
<wsdl:part name="homeStreet" type="xsd:string"/>
<wsdl:part name="homeCity" type="xsd:string"/>
<wsdl:part name="homeStateCode" type="xsd:string"/>
<wsdl:part name="homePostalCode" type="xsd:string"/>
<wsdl:part name="primaryLocation" type="xsd:string"/>
<wsdl:part name="homeLatitude" type="xsd:string"/>
<wsdl:part name="homeLongitude" type="xsd:string"/>
<wsdl:part name="shipStreet" type="xsd:string"/>
<wsdl:part name="shipCity" type="xsd:string"/>
```

```
<wsdl:part name="shipStateCode" type="xsd:string"/>
<wsdl:part name="shipPostalCode" type="xsd:string"/>
<wsdl:part name="homePhoneNumber" type="xsd:string"/>
<wsdl:part name="mobilePhoneNumber" type="xsd:string"/>
<wsdl:part name="emailAddress" type="xsd:string"/>
<wsdl:part name="hireDate" type="xsd:string"/>
<wsdl:part name="termDate" type="xsd:string"/>
<wsdl:part name="badgeId" type="xsd:string"/>
<wsdl:part name="companyCode" type="xsd:string"/>
<wsdl:part name="jobCode" type="xsd:string"/>
<wsdl:part name="departmentCode" type="xsd:string"/>
<wsdl:part name="regionCode" type="xsd:string"/>
<wsdl:part name="birthdate" type="xsd:string"/>
<wsdl:part name="lastReviewDate" type="xsd:string"/>
<wsdl:part name="payRateCode" type="xsd:string"/>
<wsdl:part name="payRate" type="xsd:string"/>
<wsdl:part name="payRateEffectiveDate" type="xsd:string"/>
<wsdl:part name="payrollId" type="xsd:string"/>
<wsdl:part name="payrollTransferFlag" type="xsd:string"/>
<wsdl:part name="payrollLocationCode" type="xsd:string"/>
<wsdl:part name="payrollGroupCode" type="xsd:string"/>
<wsdl:part name="staffTypeCode" type="xsd:string"/>
<wsdl:part name="maxHourWeek" type="xsd:string"/>
<wsdl:part name="maxHourDay" type="xsd:string"/>
<wsdl:part name="thirdPartyFlag" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="receiveDataResponse">
<wsdl:part name="receiveDataReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getStaffMaintReportRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
<wsdl:part name="importSourceList" type="xsd:string"/>
<wsdl:part name="rowTypeList" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="staffMaintWebService">
```

```
<wsdl:operation name="getStaff" parameterOrder="authCd staffNumber">

<wsdl:input message="impl:getStaffRequest" name="getStaffRequest"/>

<wsdl:output message="impl:getStaffResponse" name="getStaffResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="receiveData" parameterOrder="authCd rowType staffNumber
firstName lastName supervisorKey homeStreet homeCity homeStateCode
homePostalCode primaryLocation homeLatitude homeLongitude shipStreet shipCity
shipStateCode shipPostalCode homePhoneNumber mobilePhoneNumber emailAddress
hireDate termDate badgeId companyCode jobCode departmentCode regionCode
birthdate lastReviewDate payRateCode payRate payRateEffectiveDate payrollId
payrollTransferFlag payrollLocationCode payrollGroupCode staffTypeCode
maxHourWeek maxHourDay thirdPartyFlag">

<wsdl:input message="impl:receiveDataRequest" name="receiveDataRequest"/>

<wsdl:output message="impl:receiveDataResponse" name="receiveDataResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="getStaffMaintReport" parameterOrder="authCd startDate
endDate importSourceList rowTypeList">

<wsdl:input message="impl:getStaffMaintReportRequest"
name="getStaffMaintReportRequest"/>

<wsdl:output message="impl:getStaffMaintReportResponse"
name="getStaffMaintReportResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="staffMaintWebService.cfcSoapBinding"
type="impl:staffMaintWebService">

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="getStaff">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="getStaffRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getStaffResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
```

```
</wsdl:output>
<wsdl:fault name="CFCInvocationException">
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="receiveData">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="receiveDataRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
</wsdl:input>
<wsdl:output name="receiveDataResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
</wsdl:output>
<wsdl:fault name="CFCInvocationException">
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getStaffMaintReport">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getStaffMaintReportRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
</wsdl:input>
<wsdl:output name="getStaffMaintReportResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
</wsdl:output>
<wsdl:fault name="CFCInvocationException">
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="staffMaintWebServiceService">
```

```
<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Staff
Maintenance Web service component</wsdl:documentation>

<wsdl:port binding="impl:staffMaintWebService.cfcSoapBinding"
name="staffMaintWebService.cfc">

<wsdlsoap:address loc-
ation="http://my.naturalinsight.com/staffMaintWebService.cfc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

# Staff Maintenance Data Descriptions

Access the Staff Maintenance XML-based WSDL at https://my.naturalinsight.com/staffMaintWebService.cfc?wsdl.

The *receiveData* method of the Staff Maintenance web service adds or updates a staff member record and accepts the following data:

| Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| n/a | *authCd* | Required | | | Authorization code. Obtain from Natural Insight Client Services. |
| Type | *rowType* | Required | Char | 10 | Must be "new" "update" or "term" |
| Key | *staffNumber* | Required | Char | 25 | This value is used to uniquely identify the staff member in Natural Insight. |
| First Name | *firstName* | Required | Char | 100 | |
| Last Name | *lastName* | Required | Char | 100 | |
| Supervisor Key | *supervisorKey* | Required | Char | 25 | This value is the staff number of the staff member's supervisor. If this value is unknown, while it is not recommended, a value of "0" may be used. |
| Street Address | *homeStreet* | Optional | Char | 500 | All home street address information is to be contained in this field. There is no other home street address field. |
| City | *homeCity* | Optional | Char | 200 | |
| State | *homeStateCode* | Optional | Char | 2 | Home US 2-letter state code |
| Postal Code | *homePostalCode* | Optional | Char | 15 | Home 5 or 9 digit US zip code |
| Primary Location | *primaryLocation* | Optional | Char | 25 | (if applicable) Valid location ID of the primary location where the staff member works on a regular basis. This value is applicable to staff members who work in retail. |
| Latitude | *homeLatitude* | Optional | Float | 12 | (if available) Valid home latitude in decimal degrees. Latitude measurements range from 0° to (+/–)90°. |
| Longitude | *homeLongitude* | Optional | Float | 12 | (if available) Valid home longitude in decimal degrees. Longitude measurements range from 0° to (+/–)180°. |

| Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Shipping Address | *shipStreet* | Optional | Char | 500 | All shipping street address information is to be contained in this field. There is no other shipping street address field. |
| Shipping City | *shipCity* | Optional | Char | 200 | |
| Shipping State | *shipStateCode* | Optional | Char | 2 | Shipping US 2-letter state code |
| Shipping Postal Code | *shipPostalCode* | Optional | Char | 15 | Shipping 5 or 9 digit US zip code |
| Home Phone | *homePhoneNumber* | Optional | Char | 25 | |
| Mobile Phone | *mobilePhoneNumber* | Optional | Char | 25 | |
| E-mail Address | *emailAddress* | Optional | Char | 250 | |
| Activation Date | *hireDate* | Optional | Date | 8 | Date this staff record (new or update) is to become active in Natural Insight This is usually set to the hire date. The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |
| Termination Date | *termDate* | Optional | Date | 8 | Date this staff record (new or update) is to become inactive in Natural Insight This is usually set to the termination date. The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. This element only applies to “term” Type/rowType and defaults to the processing date if no date is provided. |
| Badge ID | *badgeId* | Optional | Char | 50 | |
| Company Code | *companyCode* | Optional | Char | 50 | |
| Job Code | *jobCode* | Optional | Char | 50 | |
| Department Code | *departmentCode* | Optional | Char | 50 | |

| Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Region Code | *regionCode* | Optional | Char | 50 | |
| Birthdate | *birthdate* | Optional | Date | 8 | The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |
| Last Review Date | *lastReviewDate* | Optional | Date | 8 | The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |
| Pay Rate Code | *payRateCode* | Optional | Char | 25 | |
| Pay Rate | *payRate* | Optional | Float | 9 | Do not include the currency sign (such as $ for American dollar). Decimal place is optional but will default to ".00" if not sup-plied. |
| Pay Rate Effective Date | *payRateEffectiveDate* | Optional | Date | 8 | The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |
| Payroll ID | *payrollId* | Optional | Char | 50 | |
| Payroll Transfer Flag | *payrollTransferFlag* | Optional | Char | 10 | |
| Payroll Loca-tion Code | *payrollLocationCode* | Optional | Char | 25 | |
| Payroll Group Code | *payrollGroupCode* | Optional | Char | 25 | |
| Staff Type Code | *staffTypeCode* | Required | Char | 5 | Must be "mgr1" "mgr2" "mgr3" "exec" "pm" or "admin". If unknown, a value of "rep" should be supplied. |
| Max Hours Weekly | *maxHourWeek* | Optional | | | |
| Max Hours Daily | *maxHourDay* | Optional | | | |
| Third Party | *thirdPartyFlag* | Optional | | | |

# Staff Maintenance Web Service Methods

The Staff Maintenance web service can be consumed using either the SOAP protocol or simple HTTP GET/POST requests. There are three public methods in the Staff Maintenance web service: *receiveData* to perform a staff member add or update operation (IMPORT), *getStaff* to read a staff member's data (EXPORT) and *getStaffMaintReport* to report on staff maintenance operations (EXPORT).

## The receiveData Method

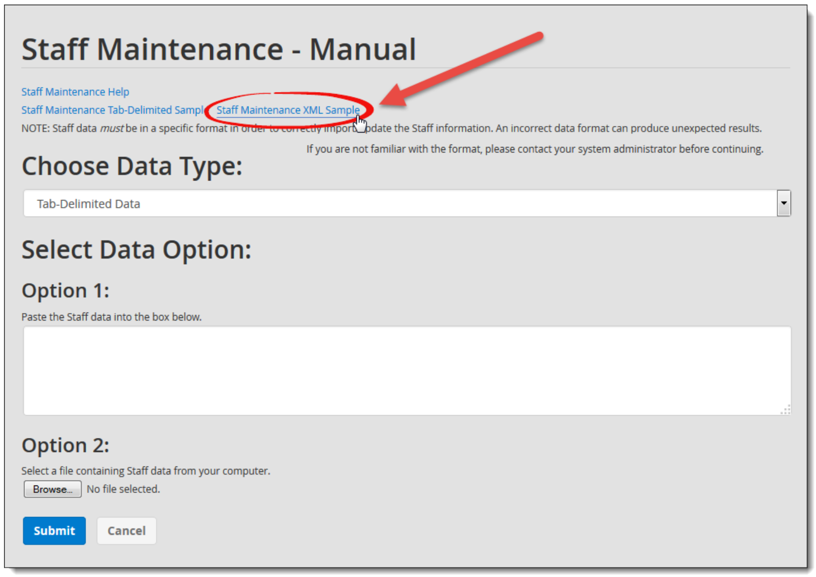
The *receiveData* method in the Staff Maintenance web service performs either a remote add or update of a staff member in the NI platform.

### Process for Calling the receiveData Staff Maintenance Method

The general process for calling the *receiveData* process is as follows:

1. An NI Administrator at your company requests from Natural Insight Client Services a unique authorization code (*authCd*) for your client instance of Natural Insight to access and employ Natural Insight Web Services.

2. If needed, the NI Administrator, who will have permissions to the Staff Maintenance interface (**STAFF > Staff Maintenance**), can obtain a sample of the Staff Maintenance XML attributes expected by the *receiveData* method by clicking on the *Staff Maintenance XML Sample* hyperlink.



Staff Maintenance - Manual

Staff Maintenance Help
Staff Maintenance Tab-Delimited Sample Staff Maintenance XML Sample
NOTE: Staff data *must* be in a specific format in order to correctly import/update the Staff information. An incorrect data format can produce unexpected results.
If you are not familiar with the format, please contact your system administrator before continuing.

Choose Data Type:
Tab-Delimited Data

Select Data Option:

Option 1:
Paste the Staff data into the box below.

Option 2:
Select a file containing Staff data from your computer.
Browse... No file selected.

Natural Insight recommends obtaining the latest version of the Staff Maintenance XML data structure from this sample in order to ensure you are aware of all the current required and optional XML data for Staff Maintenance in Natural Insight.

The sample XML is as follows:

```
</root>
   <staff rowType="" staffNumber="" firstName="" lastName=""
   supervisorKey="" homeStreet="" homeCity="" homeStateCode=""
   homePostalCode="" homeCountryCode="" primaryLocation=""
   homeLatitude="" homeLongitude="" shipStreet="" shipCity=""
   shipStateCode="" shipPostalCode="" shipCountryCode=""
   homePhoneNumber="" mobilePhoneNumber="" emailAddress="" hireDate=""
   termDate="" badgeId="" companyCode="" jobCode="" departmentCode=""
   regionCode="" birthdate="" lastReviewDate="" payRateCode=""
   payRate="" payRateEffectiveDate="" payrollId=""
   payrollTransferFlag="" payrollLocationCode="" payrollGroupCode=""
   staffTypeCode="" maxHourDay="" maxHourWeek="" thirdParty=""/>
</root>
```

3. The client programs code to pass in a single staff member's data at a time according to the XML data structure received from the **Staff Maintenance** interface by clicking on the *Staff Maintenance XML Sample*.

4. In the code, the client system calls the Staff Maintenance web service's *receiveData* method passing in the required parameters.

```
<wsdl:operation name="receiveData" parameterOrder="authCd rowType
staffNumber firstName lastName supervisorKey homeStreet homeCity
homeStateCode homePostalCode primaryLocation homeLatitude homeLongitude
shipStreet shipCity shipStateCode shipPostalCode homePhoneNumber
mobilePhoneNumber emailAddress hireDate termDate badgeId companyCode
jobCode departmentCode regionCode birthdate lastReviewDate payRateCode
payRate payRateEffectiveDate payrollId payrollTransferFlag payrollLoca-
tionCode payrollGroupCode staffTypeCode maxHourWeek maxHourDay
thirdPartyFlag">
```

## *receiveData Parameters*

1. Authorization code as a string (*authCd*)

2. XML staff parameters:

   - *rowType*
   - *staffNumber*
   - *firstName*
   - *lastName*
   - *supervisorKey*
   - *homeStreet*
   - *homeCity*
   - *homeStateCode*
   - *homePostalCode*
   - *homeCountryCode*
   - *primaryLocation*

- *homeLatitude*
- *homeLongitude*
- *shipStreet*
- *shipCity*
- *shipStateCode*
- *shipPostalCode*
- *shipCountryCode*
- *homePhoneNumber*
- *mobilePhoneNumber*
- *emailAddress*
- *hireDate*
- *termDate*
- *badgeId*
- *companyCode*
- *jobCode*
- *departmentCode*
- *regionCode*
- *birthdate*
- *lastReviewDate*
- *payRateCode*
- *payRate*
- *payRateEffectiveDate*

- *payrollId*
- *payrollTransferFlag*
- *payrollLocationCode*
- *payrollGroupCode*
- *staffTypeCode*
- *maxHourDay*
- *maxHourWeek*
- *thirdParty*

***IMPORTANT NOTES ON XML STAFF PARAMETERS***

a. The XML staff parameters passed in for each call to the *receiveData* method will process a single import - ADD or UPDATE - staff person record at a time.

b. The XML staff parameters are required - i.e. they must be passed in - but may contain empty string values for the referenced attributes.

5. If the parameters are valid, Natural Insight attempts to add or update staff member by staff member using the XML parameters provided.

6. Whether the operation is successful or not, the results of the remote add or update can be viewed via report by using the **Staff Maintenance Reports** interface (**STAFF > Staff Maintenance Reports**) or the *getStaffMaintReport* method of the Staff Maintenanceweb service.

7. An operation can be successful [1] without any warnings or [2] with a warning:

### *Staff Maintenance Success and Warning Messaging*

1. If the add or update (including terminate) operation was successful with no warning, *"OK"* is returned.

```
<return>OK</return>
```

2. If the add or update (including terminate) operation was successful but a warning is thrown, the two situations warranting warnings include:

    - When the supervisor key used is 0, the staff record will be created or updated with 0 as the staff's supervisor key.

      ```
      <return>Warning: Supervisor Key set to zero.</return>
      ```

    - When an invalid Supervisor Key provided does not match any supervisor key record, the staff record will be created or updated with 0 as the staff's supervisor key.

      ```
      <return>Warning: Supervisor Key not found.</return>
      ```

8. If the ADD or UPDATE operation was not successful, potential error messages include:

### *receiveData Error Messaging*

- Invalid or missing *authCd* returns an empty string (i.e. "") for security reasons.
- *rowType* not among the accepted values of *new*, *update* or *term*:

  ```
  <return>Invalid rowType.</return>
  ```

- Staff ID (*staffNumber*) not provided:

  ```
  <return>StaffNumber is required.</return>
  ```

- First name not provided:

  ```
  <return>FirstName is required.</return>
  ```

- Last name not provided:

  ```
  <return>LastName is required.</return>
  ```

- Supervisor key not provided:

  ```
  <return>SupervisorKey is required.</return>
  ```

- Duplicate ADD staff record:

```
<return>Error: Type was set as new, but a staff record was found.
Record not processed.</return>
```

- Staff record not found to terminate:

```
<return>Error: Type was set to terminate, but no staff record was
found. Record not processed.</return>
```

- Data errors: Not returned by the service.

## The getStaff Method

The *getStaff* method is a read function for a single staff person in the NI system. It is useful for confirming that a staff member add or update by the *receiveData* method has been successfully performed.

### Process for Calling the getStaff Staff Maintenance Method

The general process for calling the *getStaff* process is as follows:

1. The client programs code that will pass in a single staff member's ID (*staffNumber*) to read the staff attributes associated with its <staff> XML element.

2. In the code, the client system calls the web service's *getStaff* method passing in the required parameters.

```
<wsdl:operation name="getStaff" parameterOrder="authCd staffNumber">
```

#### *getStaff Parameters*

1. Authorization code as a string (*authCd*)

2. Staff ID (*staffNumber*)

3. Since the *getStaff*method allows you to read the same attributes of a staff member that you can update via Staff Maintenance, the method is useful for validating staff adds and updates as well as for exporting staff data to your network of systems. The method returns the staff information in XML format. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <staff staffNumber="1080" firstName="Mary" lastName="Sharp"
   supervisorKey="1040" homeStreet="Some Special Char St. $%^&*()!@#"
   homeCity="Sterling" homeStateCode="VA" homePostalCode="20166"
   primaryLocation="0027612" homeLatitude="39.029794"
   homeLongitude="-77.409142" shipStreet="467 Mary Lane S"
   shipCity="Centreville" shipStateCode="VA" shipPostalCode="20120"
   homePhoneNumber="5174362801" mobilePhoneNumber="5174362801"
   emailAddress="support@naturalinsight.com" hireDate="" termDate=""
   badgeId="11.11" companyCode="11.11" jobCode="11.11"
   departmentCode="10.5" regionCode="" birthdate="" lastReviewDate=""
   payRateCode="11.11" payRate="15.00" payRateEffectiveDate=""
   payrollId="11.11" payrollTransferFlag=""
   payrollLocationCode="11.11" payrollGroupCode="11.11"
   staffTypeCode="rep" maxHourWeek="40" maxHourDay="8"
   thirdPartyFlag="0" />
</root>
```

## The getStaffMaintReport Method

The *getStaffMaintReport* method is a report (read) function for staff maintenance operations - adds, updates, terminations, etc. - during a specified date range. It is useful for confirming how (either via the *receiveData* method of the Staff Maintenanceweb service or manually via the **Staff Maintenance - Manual** interface ) and when a staff member add or update has been performed. Of particular use are the *userName* and *password* attributes for each staff person generated in the *getStaffMaintReport* export.

### Process for Calling the getStaffMaintReport Staff Maintenance Method

The general process for calling the *getStaffMaintReport* method is as follows:

1. The client programs code that will receive a date range during which the staff maintenance operations performed will be read (exported).

2. In the code, the client system calls the web service's *getStaffMaintReport* method passing in the required parameters. The *getStaffMaintReport* operation code follows:

```
<wsdl:operation name="getStaffMaintReport"
parameterOrder="authCd startDate endDate importSourceList rowTypeList">
   <wsdl:input message="impl:getStaffMaintReportRequest"
   name="getStaffMaintReportRequest"/>
   <wsdl:output message="impl:getStaffMaintReportResponse"
   name="getStaffMaintReportResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

#### *getStaffMaintReport Parameters*

##### Required Parameters

a. Authorization code as a string (*authCd*)

b. Start date of date range as a string (*startDate*)

c. End date of date range as a string (*endDate*)

### Optional Parameters

a. Import source as a string (*importSourceList*); possible values include:

- *Manual*, which comprises tab-delimited data or XML data uploaded to the system via the **Staff Maintenance - Manual** page interface (**STAFF > Staff Maintenance**), or
- *Web Service*, which refers to the *receiveData* method of the Staff Maintenance web service

b. Type of maintenance operation performed for staff person as a string (*rowTypeList*); possible values include:

- addition of staff = *new*,
- update of staff = *updated*
- termination of staff = *terminated*)

3. Since the *getStaffMaintReport*method allows you to read the same attributes of a staff member operation that you can view via the **Staff Maintenance Reports** page, the method is useful for reporting on staff adds (including staff *userName*s and *password*s), updates and terminations as well as for exporting this data to your systemsn etwork. The method returns the staff information in XML format with each staff maintenance operation represented as a <staffMaint> tag. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
   <staffMaint importProcessDate="2014-02-05 11:34:00"
   importSource="Manual" processResultMsg="Complete." rowType="new"
   staffNumber="232323" staffType="Merchandiser"
   street="8923 Wilson Ave" city="Manassas" stateCode="VA"
   postalCode="20110" lastName="Farrington" firstName="Amber" emailAd-
   dress="" phoneNumber="" userName="NG-U5K-AU2R"
   password="maize-walnut" recordProcessDate="2014-02-05 11:34:00"
   recordResultMsg="OK" supervisorFullName="Demo Supervisor"
   supervisorEmailAddress="support@naturalinsight.com"
   supervisorStaffNumber="1040" />
   <staffMaint importProcessDate="2014-02-05 11:34:00"
   importSource="Manual" processResultMsg="Complete." rowType="new"
   staffNumber="595959" staffType="Merchandiser"
   street="11917 Appling Valley Rd" city="Fairfax" stateCode="VA"
   postalCode="22030" lastName="Jackson" firstName="Annie"
   emailAddress="" phoneNumber="" userName="JU-U5S-X49H"
```

```
    password="cactus-lime" recordProcessDate="2014-02-05 11:34:00"
    recordResultMsg="OK" supervisorFullName="Demo Supervisor"
    supervisorEmailAddress="support@naturalinsight.com"
    supervisorStaffNumber="1040" />
</root>
```

# Staff Maintenance Web Service API Summary

## URL

https://my.naturalinsight.com/staffMaintWebService.cfc

## WSDL

https://my.naturalinsight.com/staffMaintWebService.cfc?wsdl

## Methods

### receiveData

```
<wsdl:operation name="receiveData" parameterOrder="authCd rowType staffNumber
firstName lastName supervisorKey homeStreet homeCity homeStateCode
homePostalCode primaryLocation homeLatitude homeLongitude shipStreet shipCity
shipStateCode shipPostalCode homePhoneNumber mobilePhoneNumber emailAddress
hireDate termDate badgeId companyCode jobCode departmentCode regionCode
birthdate lastReviewDate payRateCode payRate payRateEffectiveDate payrollId
payrollTransferFlag payrollLocationCode payrollGroupCode staffTypeCode
maxHourWeek maxHourDay thirdPartyFlag">
```

#### *receiveData Parameters*

1. required string *authCd*
2. required string *rowType*
3. required string *staffNumber*
4. required string *firstName*
5. required string *lastName*
6. required string *supervisorKey*
7. optional string *homeStreet*
8. optional string *homeCity*

9. optional string *homeStateCode*
10. optional string *homePostalCode*
11. optional string *primaryLocation*
12. optional floating-point number *homeLatitude*
13. optional floating-point number *homeLongitude*
14. optional string *shipStreet*
15. optional string *shipCity*
16. optional string *shipStateCode*
17. optional string *shipPostalCode*
18. optional string *homePhoneNumber*
19. optional string *mobilePhoneNumber*
20. optional string *emailAddress*
21. optional date *hireDate*
22. optional date *termDate*
23. optional string *badgeId*
24. optional string *companyCode*
25. optional string *jobCode*
26. optional string *departmentCode*
27. optional string *regionCode*
28. optional date *birthdate*
29. optional date *lastReviewDate*
30. optional string *payRateCode*
31. optional floating-point number *payRate*

32. optional date *payRateEffectiveDate*
33. optional string *payrollId*
34. optional string *payrollTransferFlag*
35. optional string *payrollLocationCode*
36. optional string *payrollGroupCode*
37. required string *staffTypeCode*
38. optional floating-point number *maxHourWeek*
39. optional floating-point number *maxHourDay*
40. optional string *thirdPartyFlag*

### *receiveData Description*

The Staff Maintenance *receiveData* method processes a single import - ADD or UPDATE - staff person record based on the required supplied parameter values. If a parameter in an UPDATE is left blank, the value of this parameter will be cleared.

### *receiveData Error Messaging*

- Invalid or missing *authCd* returns an empty string (i.e. “”) for security reasons.
- *rowType* not among the accepted values of *new*, *update* or *term*:

```
<return>Invalid rowType.</return>
```

- Staff ID (*staffNumber*) not provided:

```
<return>StaffNumber is required.</return>
```

- First name not provided:

```
<return>FirstName is required.</return>
```

- Last name not provided:

```
<return>LastName is required.</return>
```

- Supervisor key not provided:

```
<return>SupervisorKey is required.</return>
```

- Duplicate ADD staff record:

```
<return>Error: Type was set as new, but a staff record was found. Record
not processed.</return>
```

- Staff record not found to terminate:

```
<return>Error: Type was set to terminate, but no staff record was found.
Record not processed.</return>
```

- Data errors: Not returned by the service.

## getStaff

```
<wsdl:operation name="getStaff" parameterOrder="authCd staffNumber">
```

### *getStaff Parameters*

- required string *authCd*
- required string *staffNumber*

### *getStaff Description*

The *getStaff* method allows you to read the same attributes of a staff member that you can update via Staff Maintenance's *receiveData* method. *getStaff* is useful for validating staff adds and updates as well as for exporting staff data in XML format to other systems.

### *getStaff Error Messaging*

- Invalid or missing *authCd* returns an empty string (i.e. "") for security reasons.

## getStaffMaintReport

```
<wsdl:operation name="getStaffMaintReport"
parameterOrder="authCd startDate endDate importSourceList rowTypeList">
   <wsdl:input message="impl:getStaffMaintReportRequest"
   name="getStaffMaintReportRequest"/>
   <wsdl:output message="impl:getStaffMaintReportResponse"
   name="getStaffMaintReportResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

### *getStaffMaintReport Parameters*

#### Required Parameters

a. Authorization code as a string (*authCd*)

b. Start date of date range as a string (*startDate*)

c. End date of date range as a string (*endDate*)

#### Optional Parameters

a. Import source as a string (*importSourceList*); possible values include:

   - *Manual*, which comprises tab-delimited data or XML data uploaded to the system via the **Staff Maintenance - Manual** page interface (**STAFF > Staff Maintenance**), or
   - *Web Service*, which refers to the *receiveData* method of the Staff Maintenance web service

b. Type of maintenance operation performed for staff person as a string (*rowTypeList*); possible values include:

   - addition of staff = *new*,
   - update of staff = *updated*
   - termination of staff = *terminated*)

### *getStaffMaintReport Description*

The *getStaffMaintReport* method will return an XML representation of the Staff Maintenance report that can be generated via the **Staff Mantenance Report** page in the Natural Insight interface under the **STAFF** menu (**STAFF > Staff Maintenance Reports**). This report details all staff maintenance operations - additions (including issuance of usernames and passwords) of new staff members and updates of current staff members including terminations - either manually via the **Staff Maintenance - Manual** page interface or remotely via the *receiveData* method of the Staff Maintenance web service.

### *getStaffMaintReport Error Messaging*

- Invalid or missing *authCd* returns an empty string (i.e. “”) for security reasons.

Chapter 4

# Visit Creation Web Service

# Visit Creation Web Service Overview

The Visit Creation[1] web service provides clients a secure means of creating visits in Natural Insight through an API. Each call to the Visit Creation Web Service will add one visit record at a time (unlike the bulk visit creation process via spreadsheet in which several visit records across various projects can be added or updated simultaneously).

## Description

Add a visit record in real-time remotely using XML provided by Natural Insight. This import process can be performed at any time and can be triggered by a target third-party system if needed.

### Frequency

On-demand

## WSDL

You can access the WSDL for the Visit Creation API at https://my.naturalinsight.com/visitWebService.cfc?wsdl.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://DefaultNamespace" xmlns:intf="http://DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://DefaultNamespace">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">

<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType name="CFCInvocationException">

<sequence/>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

```
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="createRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="accessKey" type="xsd:string"/>
<wsdl:part name="projectId" type="xsd:string"/>
<wsdl:part name="locationId" type="xsd:string"/>
<wsdl:part name="managerRoleId" type="xsd:string"/>
<wsdl:part name="availStartDate" type="xsd:string"/>
<wsdl:part name="availEndDate" type="xsd:string"/>
<wsdl:part name="noSpecificTimeFlag" type="xsd:string"/>
<wsdl:part name="availStartTime" type="xsd:string"/>
<wsdl:part name="availEndTime" type="xsd:string"/>
<wsdl:part name="expectedTime" type="xsd:string"/>
<wsdl:part name="availDayList" type="xsd:string"/>
<wsdl:part name="resourceNum" type="xsd:string"/>
<wsdl:part name="dataReqFlag" type="xsd:string"/>
<wsdl:part name="faxReqFlag" type="xsd:string"/>
<wsdl:part name="visitNote1" type="xsd:string"/>
<wsdl:part name="managerNote1Flag" type="xsd:string"/>
<wsdl:part name="visitNote2" type="xsd:string"/>
<wsdl:part name="managerNote2Flag" type="xsd:string"/>
<wsdl:part name="visitNote3" type="xsd:string"/>
<wsdl:part name="managerNote3Flag" type="xsd:string"/>
<wsdl:part name="primaryLocation" type="xsd:string"/>
<wsdl:part name="extraEffortType" type="xsd:string"/>
<wsdl:part name="extraEffortAmount" type="xsd:string"/>
<wsdl:part name="staffNumber" type="xsd:string"/>
<wsdl:part name="schedDate" type="xsd:string"/>
<wsdl:part name="schedTime" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="createResponse">
<wsdl:part name="createReturn" type="xsd:string"/>
```

```
</wsdl:message>

<wsdl:portType name="visitWebService">

<wsdl:operation name="create" parameterOrder="authCd accessKey projectId
locationId managerRoleId availStartDate availEndDate noSpecificTimeFlag
availStartTime availEndTime expectedTime availDayList resourceNum dataReqFlag
faxReqFlag visitNote1 managerNote1Flag visitNote2 managerNote2Flag visitNote3
managerNote3Flag primaryLocation extraEffortType extraEffortAmount
staffNumber schedDate schedTime">

<wsdl:input message="impl:createRequest" name="createRequest"/>

<wsdl:output message="impl:createResponse" name="createResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="visitWebService.cfcSoapBinding"
type="impl:visitWebService">

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="create">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="createRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>

</wsdl:input>

<wsdl:output name="createResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://DefaultNamespace"
use="encoded"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="visitWebServiceService">

<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Visit Web
service component</wsdl:documentation>

<wsdl:port binding="impl:visitWebService.cfcSoapBinding"
name="visitWebService.cfc">

<wsdlsoap:address
location="https://staging.naturalinsight.com/visitWebService.cfc"/>
```

```
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

# Visit Creation Data Descriptions

Access the Visit Creation[1] XML-based WSDL at https://my.naturalinsight.com/visitWebService.cfc?wsdl.

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| n/a | *authCd* | Required | | | Authorization code. Obtain from Natural Insight Client Services. |
| n/a | *accessKey* | Required | | | Access Key. This key is unique to each of your clients who will be creating visits within Natural Insight. As a result, each of your individual clients' accessKeys must be obtained from Natural Insight Client Services. |
| Project Number | *projectId* | Required | | | The project ID of the project in which the visit will belong. |
| Location ID | *locationId* | Required | | | The location ID of the location where the visit will take place. |
| Manager Role ID | *managerRoleId* | Required | | | The manager role ID of the manager at the location where the visit will take place. |
| Earliest Date | *availStartDate* | Required | | | The earliest date the visit can be scheduled in Natural Insight The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |
| Latest Date | *availEndDate* | Required | | | The latest date the visit can be scheduled in Natural Insight. The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values. |

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| n/a[1] | *noSpecificTimeFlag* [2] | Required | | | Possible values include “true” / ”yes” / ”1” or “false” / “no” / “0”. Characters are case-insens-itive so, as an example, “YES” “Yes” “yes” “yEs” etc. would each be accepted as valid. If the value of the *noSpecificTimeFlag* element is any valid case variation of “false” “no” or “0” then the visit can be scheduled for any time of day. |
| Earliest Start Time | *availStartTime* | Required | | | The earliest time the visit can be scheduled to begin within the 24-hour period of each day within the *availStartDate* and the *availEndDate*. The time must be in the ISO 8061 format - hh:mm or hh:mm:ss - with hh, mm and ss being 2 characters, padded in front with a 0 for single-digit values. |
| Latest Start Time | *availEndTime* | Required | | | The latest time the visit can be scheduled to begin within the 24-hour period of each day within the *availStartDate* and the *availEndDate*. The time must be in the ISO 8061 format - hh:mm or hh:mm:ss - with hh, mm and ss being 2 characters, padded in front with a 0 for single-digit values. |
| Schedule Time | *expectedTime* | Required | | | The expected duration of the visit in minutes. For example, if the visit is to last one hour, “60” should be entered. The *expectedTime* number of minutes will be blocked off on the calendar of the staff member who schedules the visit for him or herself in order to indicate the duration of the visit. |
| Days of Week | *availDayList* | Required | | | The list of numeric day values (i.e., the days of the week) when the visit may be scheduled within the *availStartDate* and the *availEndDate*. “1” equals Sunday; “2” equals Monday and so on. For example, if the visit may be scheduled for any day of the week, *availDayList* = “1,2,3,4,5,6,7”. If the visit may only be scheduled Monday-Friday, *availDayList* = “2,3,4,5,6”, etc. |

[1] *If no values are included in the Earliest Start Time and Latest Start Time columns, this would be the equivalent of the noSpecificTimeFlag. However, because the spreadsheet columns do not contain a noSpecificTimeFlag, the way to indicate that a visit can be scheduled for any time of day is to simply leave the values for the Earliest Start Time and Latest Start Time columns blank.*

[2] *Need clarification here about how noSpecificTimeFlag works in conjunction with availStartTime and availEndTime.*

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Resource Number | *resourceNum* | Required | | | The number of visits to create. For example, if you indicated that *resourceNum* has a value of "2" this means that 2 identical visits will be created though each will be given a unique visit ID to distinguish them. If a *resourceNum* value of "2" or higher is used, then the *staffNumber*, *schedDate* and *schedTime* elements cannot be utlized for this visit. |
| Required Survey | *dataReqFlag* | Required | | | *dataReqFlag* indicates whether a data survey is required for the visit being created. Possible values include "true" / "yes" / "1" or "false" / "no" / "0". Characters are case-insensitive so, as an example, "YES" "Yes" "yes" "yEs" etc. would each be accepted as valid. If the visit is to require a data survey, use one of the "yes" values for *dataReqFlag*. |
| Required Fax | *faxReqFlag* | Required | | | *faxReqFlag* indicates whether a fax survey is required for the visit being created. Possible values include "true" / "yes" / "1" or "false" / "no" / "0". Characters are case-insensitive so, as an example, "YES" "Yes" "yes" "yEs" etc. would each be accepted as valid. If the visit is to require a fax survey, use one of the "yes" values for *faxReqFlag*. |
| Visit Note 1 | *visitNote1* | Optional | | | The first visit note of the visit being created. If the value of *visitNote1* is provided, it will be populated in the Visit Note 1 field of the visit being created. |
| Manager Only Flag 1 | *managerNote1Flag* | Optional | | | *managerNote1Flag* indicates whether *visitNote1* will be visible to all users (including "staff member" level users) within your client instance of Natural Insight, or to only "mgr1" or above level users. Possible values include "true" / "yes" / "1" or "false" / "no" / "0". Characters are case-insensitive so, as an example, "YES" "Yes" "yes" "yEs" etc. would each be accepted as valid. If *visitNote1* is to be visible to "mgr1" or above level users only, use one of the "yes" values for *managerNote1Flag*. |

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Visit Note 2 | *visitNote2* | Optional | | | The second visit note of the visit being created. If the value of *visitNote2* is provided, it will be populated in the Visit Note 2 field of the visit being created. |
| Manager Only Flag 2 | *managerNote2Flag* | Optional | | | *managerNote2Flag* indicates whether *visitNote2* will be visible to all users (including “staff member” level users) within your client instance of Natural Insight, or to only “mgr1” or above level users. Possible values include “true” / ”yes” / ”1” or “false” / “no” / “0”. Characters are case-insensitive so, as an example, “YES” “Yes” “yes” “yEs” etc. would each be accepted as valid. If *visitNote2* is to be visible to “mgr1” or above level users only, use one of the “yes” values for *managerNote2Flag*. |
| Visit Note 3 | *visitNote3* | Optional | | | The third visit note of the visit being created. If the value of *visitNote3* is provided, it will be populated in the Visit Note 3 field of the visit being created. |
| Manager Only Flag 3 | *managerNote3Flag* | Optional | | | *managerNote3Flag* indicates whether *visitNote3* will be visible to all users (including “staff member” level users) within your client instance of Natural Insight, or to only “mgr1” or above level users. Possible values include “true” / ”yes” / ”1” or “false” / “no” / “0”. Characters are case-insensitive so, as an example, “YES” “Yes” “yes” “yEs” etc. would each be accepted as valid. If *visitNote3* is to be visible to “mgr1” or above level users only, use one of the “yes” values for *managerNote3Flag*. |

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Primary Location | *primaryLocation* | Optional | | | Possible values for true include “yes” / “y” / “1”. Possible values for false include “no” / “n” / “0”. Characters are case-insensitive so, as an example, “YES” “Yes” “yes” “yEs” etc. would each be accepted as valid.<br><br>If (1) the location for which the visit is being created has one or more staff members who have this location designated as their Primary Location, and if (2) the *primaryLocation* element is set to true - i.e. "yes", "Y" or "1", then opportunities will be sent to these staff members to offer this visit. If the *primaryLocation* element is set to true, then the *staffNumber*, *schedDate* and *schedTime* elements cannot be used. |
| Extra Effort Type | *extraEffortType* | Optional | | | *extraEffortType* indicates the type of extra effort pay that will be provided for this visit. Possible values are “Percent of Pay” or “Stipend (Flat Rate)”. If *extraEffortType* is “Percent of Pay” or “Stipend (Flat Rate)”, then *extraEffortAmount* must be populated with a value. |
| Extra Effort Amount | *extraEffortAmount* | Optional | | | *extraEffortAmount* indicates the amount of extra effort pay the staff member who completes the visit will receive. If “25.00” were entered, this would indicate 25 dollars if “Stipend (Flat Rate)” were entered for the Extra Effort Type element. If “Percent of Pay” were entered for the Extra Effort Type element, “25.00” would indicate 25 percent of the staff member’s total pay for the visit. Values accepted for *extraEffortAmount* are decimals with 2 places such as “25.00”. The value must be less than 9999999.99.<br><br>If *extraEffortAmount* is populated with a value, then *extraEffortType* must include either “Percent of Pay” or “Stipend (Flat Rate)”. |

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Staff Number | *staffNumber* | Optional | | | The staff ID of the staff member who will be assigned and scheduled for the visit being created. If you enter a value for *staffNumber*, you must also enter valid values for both the *schedDate* and the *schedTime* elements. All three of the elements - *staffNumber*, *schedDate* and *schedTime* - must be used together in order to assign the visit being created to a staff member and then schedule it on the said staff member's calendar. When *staffNumber*, *schedDate* and *schedTime* are used together, *primaryLocation* must be blank. When *staffNumber*, *schedDate* and *schedTime* are employed together in the visit creation process, Natural Insight refers to this as importing - i.e. creating - a visit with "assignment." |
| Assignment Date | *schedDate* | Optional | | | The date the visit being created will be scheduled for the staff member whose staff ID is referenced in the *staffNumber* element. If you enter a value for *schedDate*, you must also enter valid values for both the *staffNumber* and the *schedTime* elements. All three of the elements - *staffNumber*, *schedDate* and *schedTime* - must be used together in order to assign the visit being created to a staff member and then schedule it on the said staff member's calendar. When *staffNumber*, *schedDate* and *schedTime* are used together, *primaryLocation* must be blank. When *staffNumber*, *schedDate* and *schedTime* are employed together in the visit creation process, Natural Insight refers to this as importing - i.e. creating - a visit with "assignment." |

| Tab-Delimited Field Name | XML Attribute | Required? | Data Type | Max Length | Description |
|---|---|---|---|---|---|
| Assignment Start Time | *schedTime* | Optional | | | The time the visit being created will be scheduled to start for the staff member whose staff ID is referenced in the *staffNumber* element. If you enter a value for *schedTime*, you must also enter valid values for both the *staffNumber* and the *schedDate* elements. All three of the elements - *staffNumber*, *schedDate* and *schedTime* - must be used together in order to assign the visit being created to a staff member and then schedule it on the said staff member's calendar. When *staffNumber*, *schedDate* and *schedTime* are used together, *primaryLocation* must be blank. When *staffNumber*, *schedDate* and *schedTime* are employed together in the visit creation process, Natural Insight refers to this as importing - i.e. creating - a visit with "assignment." |

# Visit Creation Web Service create Method

The Visit Creation web service can be consumed using either the SOAP protocol or simple HTTP GET/POST requests. There is only one public method in the Visit Creation web service: *create*.

## Process for Calling the create Visit Creation Method

The general process for calling the *create* process is as follows:

1. An NI Administrator at your company requests from Natural Insight Client Services a unique authorization code (*authCd*) for your client instance of Natural Insight to access and employ Natural Insight Web Services.

2. The administrator also requests from Natural Insight Client Services an access key (*accessKey*), which is a custom code created for each of your clients to whom you would like to grant access to the Natural Insight Web Services Visit Creation API and, thus, give this client the ability to create visits within your instance of Natural Insight. In this way, Natural Insight can revoke access to the client to whom you have granted access to the Visit Creation API upon your request. If you have several clients to whom you would like to grant access to the Visit Creation web service, you will request a unique access key (*accessKey*) for each one.

3. The NI Administrator can reference the Visit Creation WSDL at https://my.naturalinsight.com/visitWebService.cfc?wsdl in order to examine the parameters of the *create* operation/method.

```
<wsdl:operation name="create" parameterOrder="authCd accessKey projectId
locationId managerRoleId availStartDate availEndDate noSpecificTimeFlag
availStartTime availEndTime expectedTime availDayList resourceNum
dataReqFlag faxReqFlag visitNote1 managerNote1Flag visitNote2
managerNote2Flag visitNote3 managerNote3Flag primaryLocation
extraEffortType extraEffortAmount staffNumber schedDate schedTime">
```

4. The client programs code to pass a single visit ’s data at a time according to the XML data structure described in the <wsdl:operation> tag of the *create* method.

5. In the code, the client system calls the Visit Creation web service's *create* method passing in the required parameters:

    1. Authorization code as a string (*authCd*)

    2. Access key as a string (*accessKey*)

    3. Other XML visit parameters for the *create* method:

        - Project ID (*projectId*)
        - Location ID (*locationId*)
        - Manager Role ID (*managerRoleId*)
        - Earliest Start Date (*availStartDate*)
        - Latest Start Date (*availEndDate*)
        - No Specific Time Indicated (*noSpecificTimeFlag*)
        - Earliest Start Time (*availStartTime*)
        - Latest Start Time (*availEndTime*)
        - Scheduled Time Duration (*expectedTime*)
        - Days of the Week Visit Possible (*availDayList*)
        - Resource Number (*resourceNum*)
        - Required Survey (*dataReqFlag*)
        - Required Fax (*faxReqFlag*)
        - Visit Note 1 (*visitNote1*)
        - Visit Note 1 for Managers Only (*managerNote1Flag*)
        - Visit Note 2 (*visitNote2*)
        - Visit Note 2 for Managers Only (*managerNote2Flag*)
        - Visit Note 3 (*visitNote3*)

- Visit Note 3 for Managers Only (*managerNote3Flag*)
- Primary Location (*primaryLocation*)
- Extra Effort Type (*extraEffortType*)
- Extra Effort Amount (*extraEffortAmount*)
- Staff ID (*staffNumber*)
- Visit Assignment Date (*schedDate*)
- Visit Assignment Start Time (*schedTime*)

***IMPORTANT NOTES ON XML VISIT CREATION PARAMETERS***

a. The XML visit parameters passed in for each call to the *create* method will ADD a single visit record at a time.

b. The XML visit parameters are required - i.e. they must be passed in - but may contain empty string values for the optional attributes.

6. If the parameters are valid, Natural Insight attempts to add a visit using the XML parameters provided.

7. If the create (ADD) visit operation was not successful, potential error messages include:

   - Bad or missing *authCd*:

```
<return>Upload Failed: Authentication Failure.</return>
```

   - Data errors: Not returned by the service.

naturalinsight®

# Visit Creation Web Service API Summary

## URL

https://my.naturalinsight.com/visitWebService.cfc

## WSDL

https://my.naturalinsight.com/visitWebService.cfc?wsdl

## Method

### create

```
<wsdl:operation name="create" parameterOrder="authCd accessKey projectId
locationId managerRoleId availStartDate availEndDate noSpecificTimeFlag
availStartTime availEndTime expectedTime availDayList resourceNum dataReqFlag
faxReqFlag visitNote1 managerNote1Flag visitNote2 managerNote2Flag visitNote3
managerNote3Flag primaryLocation extraEffortType extraEffortAmount
staffNumber schedDate schedTime">
```

#### *create Parameters*

1. required string *authCd*
2. required string *accessKey*
3. required string *projectId*
4. required string *locationId*
5. required string *managerRoleId*
6. required string *availStartDate*
7. required string *availEndDate*
8. required string *noSpecificTimeFlag*
9. required string *availStartTime*

10. required string *availEndTime*
11. required string *expectedTime*
12. required string *availDayList*
13. required string *resourceNum*
14. required string *dataReqFlag*
15. required string *faxReqFlag*
16. optional string *visitNote1*
17. optional string *managerNote1Flag*
18. optional string *visitNote2*
19. optional string *managerNote2Flag*
20. optional string *visitNote3*
21. optional string *managerNote3Flag*
22. optional string *primaryLocation*
23. optional string *extraEffortType*
24. optional string *extraEffortAmount*
25. optional string *staffNumber*
26. optional string *schedDate*
27. optional string *schedTime*

### *create Description*

The Visit Creation *create* method processes a single import ADD visit record based upon the supplied parameter values. If a parameter is left blank, the value of this parameter will not be specified.

### *Potential Error Messages*

- Bad or missing *authCd*:

```
<return>Upload Failed: Authentication Failure.</return>
```

- Data errors: Not returned by the service.

Chapter 5

# Single Sign On Web Service

# Single Sign On (Remote Access) Web Service Overview

## Description

The Natural Insight Single Sign On web service provides Natural Insight clients with the ability to authenticate their end-users into Natural Insight behind-the-scenes once these users have already signed in to another client system. In this way, when the end-users proceed to Natural Insight, from a linked client system that has already authenticated their credentials, they will enter the Natural Insight platform directly with out an additional login prompt.

The Single Sign On web service allows clients to build internal login pages that will sign in their users to Natural Insight without the users having to enter their username and password in the login page at my.naturalinsight.com.

### Frequency

On-demand

## WSDL

You can access the WSDL for the Single Sign On API at https://my.naturalinsight.com/ssoWebService.cfc?wsdl.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://niWeb" xmlns:intf="http://niWeb"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://niWeb">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">

<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType name="CFCInvocationException">

<sequence/>

</complexType>
```

```
</schema>
</wsdl:types>
<wsdl:message name="getSsoTokenResponse">
<wsdl:part name="getSsoTokenReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="loginRedirectResponse"></wsdl:message>
<wsdl:message name="loginRedirectRequest">
<wsdl:part name="ssoKey" type="xsd:string"/>
<wsdl:part name="ssoToken" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getSsoTokenRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="ssoKey" type="xsd:string"/>
<wsdl:part name="staffNumber" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="ssoWebService">
<wsdl:operation name="getSsoToken" parameterOrder="authCd ssoKey
staffNumber">
<wsdl:input message="impl:getSsoTokenRequest" name="getSsoTokenRequest"/>
<wsdl:output message="impl:getSsoTokenResponse" name="getSsoTokenResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
<wsdl:operation name="loginRedirect" parameterOrder="ssoKey ssoToken">
<wsdl:input message="impl:loginRedirectRequest" name="loginRedirectRequest"/>
<wsdl:output message="impl:loginRedirectResponse"
name="loginRedirectResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ssoWebService.cfcSoapBinding" type="impl:ssoWebService">
<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getSsoToken">
```

```
<wsdlsoap:operation soapAction=""/>

<wsdl:input name="getSsoTokenRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getSsoTokenResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="loginRedirect">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="loginRedirectRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="loginRedirectResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="ssoWebServiceService">

<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Single
Sign-On (SSO) Web service component</wsdl:documentation>

<wsdl:port binding="impl:ssoWebService.cfcSoapBinding"
name="ssoWebService.cfc">

<wsdlsoap:address
location="https://my.naturalinsight.com/ssoWebService.cfc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

# Single Sign On Web Service Methods

The Natural Insight Single Sign On web service can be consumed using either the SOAP protocol or simple HTTP GET/POST requests over SSL only. The Single Sign On web service must be accessed over SSL and failure to do so will result in an error message. There are two public methods in the Single Sign On web service: *getSsoToken* and *loginRedirect*.

## Requirements for the Single Sign On Web Service

In order for a client's end-user to successfully sign on to Natural Insight via Single Sign On, several criteria must be met.

- The Natural Insight Single Sign On web service must be accessed over SSL.
- The Natural Insight client must be active and have SSO enabled.
- The client end-user must have SSO enabled.
- The client end-user must be an active Natural Insight user (not terminated).

## Process for Calling the getSsoToken Method

The general process for working with the *getSsoToken* method is as follows:

1. An NI Administrator at your company requests from Natural Insight Client Services a unique authorization code (*authCd*) for your client instance of Natural Insight to access and employ Natural Insight Web Services.
2. The administrator also requests from Natural Insight Client Services a unique Single Sign On key (*ssoKey*), which also identifies the client, as does the *authCd*, but the *ssoKey* may be exposed to users in a URL, for example. Natural Insight can change or deactivate the *ssoKey*, should it become necessary due to security reasons. In this manner, only the Single Sign On web service would be affected while any other Natural Insight Web Services would be untouched.

3. The NI Administrator can reference the Single Sign On WSDL at https://my.naturalinsight.com/ssoWebService.cfc?wsdl in order to examine the parameters of the *getSsoToken* operation/method.

```
<wsdl:operation name="getSsoToken" parameterOrder="authCd ssoKey
staffNumber">
   <wsdl:input message="impl:getSsoTokenRequest"
   name="getSsoTokenRequest"/>
   <wsdl:output message="impl:getSsoTokenResponse"
   name="getSsoTokenResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

4. The client programs code to pass a single user's sign on data at a time according to the XML data structure described in the <wsdl:operation> tag of the *getSsoToken* method.

5. In the code, the client system will call the Single Sign On web service's *getSsoToken* method passing in the three required parameters:

    1. Authorization code as a string (*authCd*)
    2. Single Sign On key as a string (*ssoKey*)
    3. The staff ID of the end-user (*staffNumber*)

6. If the parameters are valid, the getSsoToken method returns a *ssoToken* - a UUID (Universally Unique Identifier) used to identify that user in the SSO process. After each successful login for a user, the *ssoToken* value is updated with a new UUID to ensure that exposed login URLs (in a book-mark, for example) cannot be used more than once. If the user's staff ID (*staffNumber*) is invalid, "00000000-0000-0000-0000000000000000" is returned for the *ssoToken*.

7. If the getSsoToken operation was not successful, potential error messages include:

    - Error: HTTPS required
    - Error: Invalid *ssoKey*
    - Error: Invalid *authCd*
    - Error: *staffNumber* is required

8. The client system will then need to save the NI *ssoToken* for the user who signed in in the user's session to be inserted in the *loginRedirect* method when the user wishes to log in to Natural Insight.

## Process for Calling the loginRedirect Method

The general process for working with the *loginRedirect* process is as follows:

1. The client programs code to pass in the *ssoKey* and *ssoToken* of the user who wishes to sign in to Natural Insight to the *loginRedirect* method.

2. The NI Administrator can reference the Single Sign On WSDL at https://my.naturalinsight.com/ssoWebService.cfc?wsdl in order to examine the parameters of the *loginRedirect* operation/method.

```
<wsdl:operation name="loginRedirect" parameterOrder="ssoKey ssoToken">
   <wsdl:input message="impl:loginRedirectRequest"
   name="loginRedirectRequest"/>
   <wsdl:output message="impl:loginRedirectResponse"
   name="loginRedirectResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

3. If the user is successfully authenticated, the request will perform an HTTP redirect to the user's Natural Insight Home Page. The redirect is probably most easily accomplished with a simple HTTP GET request (a URL link) that the user can click on, for example,

   https://my.naturalinsight.com/ssoWebService.cfc?method=loginRedirect&
   ssoKey=XXX-XXX-XXXX&ssoToken=XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXX.

4. The two required parameters of the *loginRedirect* method are:

   1. Single Sign On key as a string (*ssoKey*)

   2. Single Sign On token, returned from the *getSsoToken* method, as a string (*ssoToken*)

5. If the authentication fails, the user will be redirected to the Natural Insight login screen.

## Single Sign On Process Overview from a User's Standpoint

1. The user signs in to the client system and is properly authenticated.
2. The client system calls the Natural Insight Single Sign On web service's *getSsoToken* method passing in the user's staff ID (*staffNumber*), and the required keys: *authCd* and *ssoKey*.
3. If the user is valid, the *getSsoToken* method returns the *ssoToken* for that user.
4. The client system saves the *ssoToken* in the user's session to be invoked when the user wishes to sign in to Natural Insight (via a link, for example).
5. To sign in to Natural Insight, the *ssoKey* and *ssoToken* values are passed to the Single Sign On web service's *loginRedirect* method. If the user is successfully authenticated, the request will perform an HTTP redirect to the user's Natural Insight Home Page.

# Single Sign On Web Service API Summary

## URL

https://my.naturalinsight.com/ssoWebService.cfc

## WSDL

https://my.naturalinsight.com/ssoWebService.cfc?wsdl

## Methods

### getSsoToken Method

```
<wsdl:operation name="getSsoToken" parameterOrder="authCd ssoKey
staffNumber">
      <wsdl:input message="impl:getSsoTokenRequest"
      name="getSsoTokenRequest"/>
      <wsdl:output message="impl:getSsoTokenResponse"
      name="getSsoTokenResponse"/>
      <wsdl:fault message="impl:CFCInvocationException"
      name="CFCInvocationException"/>
</wsdl:operation>
```

#### *getSsoToken Parameters*

1. required string *authCd*
2. required string *ssoKey*
3. required string *staffNumber*

#### *getSsoToken Description*

The Single Sign On *getSsoToken* method returns the *ssoToken* (UUID) for the given *staffNumber* (staff ID) if the request is successful. If the user's *staffNumber* is invalid, "00000000-0000-0000-0000000000000000" is returned for the *ssoToken*.

### *getSsoToken Potential Error Messages*

- Error: HTTPS required
- Error: Invalid *ssoKey*
- Error: Invalid *authCd*
- Error: *staffNumber* is required

## loginRedirect Method

```
<wsdl:operation name="loginRedirect" parameterOrder="ssoKey ssoToken">
      <wsdl:input message="impl:loginRedirectRequest"
      name="loginRedirectRequest"/>
      <wsdl:output message="impl:loginRedirectResponse"
      name="loginRedirectResponse"/>
      <wsdl:fault message="impl:CFCInvocationException"
      name="CFCInvocationException"/>
</wsdl:operation>
```

### *loginRedirect Parameters*

1. required string *ssoKey*
2. required string *ssoToken*

### *loginRedirect Description*

The Single Sign On *loginRedirect* method performs a login operation for the given *ssoToken* and, if successful, performs an HTTP redirect to the Natural Insight Home Page of the user who has been logged in.

### *loginRedirect Potential Error Behavior*

If the *loginRedirect* authentication fails, the method will perform an HTTP redirect to the Natural Insight login screen.

# Chapter 6

# EXPORT WEB SERVICES

Chapter 6

# Natural Insight Packager Web Service

# Packager Web Service Overview

The Natural Insight Packager web service exports visit-level[1] Natural Insight XML data including project, visit, survey and timekeeping information for your client instance of the platform. You can request this data according to the following search criteria:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

## Description

Retrieve and use Natural Insight visit-level data by date range, project ID or survey ID plus date range.

***EXAMPLE OF XML HIERARCHY RETURNED WHEN XML PACKAGER DATA IS REQUESTED:***

Note: Though the XML data structure below is an example of the XML returned upon a request according to a project ID, the hierarchy of elements and sub-elements will be the same when the request is by date range or by survey ID and date range.

```
<?xml version="1.0" encoding="utf-8"?>
<root>
   <project/>
   <visit>
      <location/>
      <staff/>
   </visit>
   <visit>
      <location/>
      <staff/>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

```
</visit>
<survey type="Project-Based">
   <question>
      <choice/>
      <choice/>
   </question>
   <question></question>
   <question></question>
   <question>
      <gridQuestion></gridQuestion>
      <gridQuestion>
         <choice/>
         <choice/>
      </gridQuestion>
      <gridQuestion>
         <choice/>
         <choice/>
         <choice/>
         <choice/>
      </gridQuestion>
   </question>
</survey>
<survey type="Timekeeping">
   <question></question>
</survey>
<data>
   <response/>
   <response/>
</data>
<data>
   <response/>
   <response/>
   <response/>
   <response/>
```

```
        </data>
        <data>
            <response/>
            <response/>
        </data>
        <data>
            <response/>
            <response/>
            <response/>
            <response/>
        </data>
        <timekeepingData>
            <punch>
                <response/>
            </punch>
        </timekeepingData>
        <timekeepingData>
            <punch>
                <response/>
            </punch>
        </timekeepingData>
        <timekeepingData>
            <punch>
                <response/>
            </punch>
        </timekeepingData>
        <timekeepingData>
            <punch>
                <response/>
            </punch>
        </timekeepingData>
    </root>
```

Each XML packet of the Natural Insight Packager web service is organized into project information, visit information (containing location information about where the visit was performed and staff information about the staff person assigned to the visit), survey information, survey response information and time-keeping information. Refer to the data description tables for each category below to discover each element and its attributes.

- Project data (visit level)
- Visit data
    - Location data (visit level)
    - Staff data (visit level)
- Survey data (visit level)
- Survey response data (visit level)
- Timekeeping data (visit level)

Note: The Natural InsightPackager web service will also retrieve deleted visit data. See **Packager Deleted Visits Web Service Overview**

### Frequency

On-demand

## WSDL

You can access the WSDL for the Natural Insight Packager API at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://niWeb" xmlns:intf="http://niWeb"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://niWeb">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">

<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType name="CFCInvocationException">

<sequence/>
```

```
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="requestPackageResponse">
<wsdl:part name="requestPackageReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageBySurveyIdAndDateResponse">
<wsdl:part name="requestPackageBySurveyIdAndDateReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByProjectNumberRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="projectNumber" type="xsd:double"/>
</wsdl:message>
<wsdl:message name="requestPackageDeletedVisitByDateResponse">
<wsdl:part name="requestPackageDeletedVisitByDateReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="requestPackageRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="projectNumber" type="xsd:double"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByProjectNumberResponse">
<wsdl:part name="requestPackageByProjectNumberReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageBySurveyIdAndDateRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="surveyId" type="xsd:double"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getStatusRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="packageUUID" type="xsd:string"/>
```

```
</wsdl:message>
<wsdl:message name="getStatusResponse">
<wsdl:part name="getStatusReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getPackageResponse">
<wsdl:part name="getPackageReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageDeletedVisitByDateRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getPackageRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="packageUUID" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByDateRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByDateResponse">
<wsdl:part name="requestPackageByDateReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="packagerWebService">
<wsdl:operation name="getPackage" parameterOrder="authCd packageUUID">
<wsdl:input message="impl:getPackageRequest" name="getPackageRequest"/>
<wsdl:output message="impl:getPackageResponse" name="getPackageResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
<wsdl:operation name="getStatus" parameterOrder="authCd packageUUID">
<wsdl:input message="impl:getStatusRequest" name="getStatusRequest"/>
<wsdl:output message="impl:getStatusResponse" name="getStatusResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
```

```
<wsdl:operation name="requestPackageByDate" parameterOrder="authCd startDate
endDate">

<wsdl:input message="impl:requestPackageByDateRequest"
name="requestPackageByDateRequest"/>

<wsdl:output message="impl:requestPackageByDateResponse"
name="requestPackageByDateResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackageBySurveyIdAndDate"
parameterOrder="authCd surveyId startDate endDate">

<wsdl:input message="impl:requestPackageBySurveyIdAndDateRequest"
name="requestPackageBySurveyIdAndDateRequest"/>

<wsdl:output message="impl:requestPackageBySurveyIdAndDateResponse"
name="requestPackageBySurveyIdAndDateResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackageByProjectNumber" parameterOrder="authCd
projectNumber">

<wsdl:input message="impl:requestPackageByProjectNumberRequest"
name="requestPackageByProjectNumberRequest"/>

<wsdl:output message="impl:requestPackageByProjectNumberResponse"
name="requestPackageByProjectNumberResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackage" parameterOrder="authCd projectNumber
startDate endDate">

<wsdl:input message="impl:requestPackageRequest"
name="requestPackageRequest"/>

<wsdl:output message="impl:requestPackageResponse"
name="requestPackageResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackageDeletedVisitByDate"
parameterOrder="authCd startDate endDate">

<wsdl:input message="impl:requestPackageDeletedVisitByDateRequest"
name="requestPackageDeletedVisitByDateRequest"/>

<wsdl:output message="impl:requestPackageDeletedVisitByDateResponse"
name="requestPackageDeletedVisitByDateResponse"/>
```

```
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="packagerWebService.cfcSoapBinding"
type="impl:packagerWebService">

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="getPackage">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="getPackageRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getPackageResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="getStatus">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="getStatusRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getStatusResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageByDate">

<wsdlsoap:operation soapAction=""/>
```

```
<wsdl:input name="requestPackageByDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageByDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageBySurveyIdAndDate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageBySurveyIdAndDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageBySurveyIdAndDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageByProjectNumber">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageByProjectNumberRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageByProjectNumberResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">
```

```
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackage">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageDeletedVisitByDate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageDeletedVisitByDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageDeletedVisitByDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="packagerWebServiceService">

<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Natural
Insight packager Web service component</wsdl:documentation>

<wsdl:port binding="impl:packagerWebService.cfcSoapBinding"
name="packagerWebService.cfc">
```

```
<wsdlsoap:address
location="http://my.naturalinsight.com/packagerWebService.cfc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

# Packager Project Data Descriptions (Visit Level)

Natural Insight Packager can export XML project data (Visit Level[1]) regarding project dates, management, billing and contact information for active projects according to:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <project> contains all of the project data attributes.

***EXAMPLE OF THE <PROJECT> ELEMENT***

```
<project number="6" bundledScheduling="Yes"
createDate="2014-03-13 12:05:22.857"
updateDate="2014-09-15 11:49:52.967" title="Daily Reset"
shortTitle="Daily Reset" shortDescription="Daily reset merch"
fullDescription="Reset clothing and accessory merchandise 1x/day"
organizationName="Calvin Klein" organizationCode="277"
startDate="01/01/2014" endDate="12/31/2014"
expectedTimeMin="60" expectedTimeMax="60"
projectScheduleTime="60" projectManagerStaffNumber="0001"
projectManagerFirstName="Gordon" projectManagerLastName="Go" status-
s="Active" continuousCoverageFlag="false"
billingType="hour" billingRate="0.00"
billingMinimumMinutes="0" billingAuthorizationCode=""
billingProcessingCode="" payable="No" payType="visit"
payRate="20.00" payMinimumMinutes="0" phoneSurveyFlag="false"
webSurveyFlag="true" checkOutFlag="true" checkInFlag="false"
checkOutTimekeepingFlag="false" ivrPhoneNumber="" ivrVoiceSiteId="0"
ivrPageNumber="1" faxRequiredFlag="false" faxNumber="" faxCer-
tificationStatement="" surveyId="9" tagIdList=""/>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

**Element: <project>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *Number* | Integer | | Project ID that uniquely identifies a project |
| *bundledScheduling* | Option (Yes, No) | | Flag indicating whether the project is a Bundled Projector not |
| *createDate* | Date | | Date the project was created |
| *updateDate* | Date | | Date when any information on the project was last updated |
| *Title* | Char | 500 | Title of the project |
| *shortTitle* | Char | 150 | Short title of the project |
| *shortDescription* | Char | 150 | Short description of the project |
| *fullDescription* | Char | 3000 | Full description of the project |
| *organizationName* | Char | 50 | Name of the organization (client) tied to the pro-ject |
| *organizationCode* | Char | 150 | Organization code |
| *startDate* | Date | | Start date of project |
| *endDate* | Date | | End date of project |
| *expectedTimeMin* | Integer | | Minimum number of minutes expected to com-plete |
| *expectedTimeMax* | Integer | | Maximum number of minutes expected to com-plete |
| *projectManagerStaffNumber* | Char | 25 | Project manager's unique staff ID |
| *projectManagerFirstName* | Char | 100 | Project manager's first name |
| *projectManagerLastName* | Char | 100 | Project manager's last name |
| *Status* | Option (active, inact-ive, pending, deleted, unknown) | | Status of the project |
| *continuousCoverageFlag* | Option (true, false) | | Flag indicating continuous coverage |

| Element: <project> | | | |
|---|---|---|---|
| XML Attribute | Data Type | Max Data Size | Description |
| *billingType* | Option (hour, visit) | | Billing type indicating whether this project is billed per visit or per hour |
| *billingRate* | Decimal | (9, 2) | Billing rate per visit or per hour (depending on the value of billingType) |
| *billingMinimumMinutes* | Integer | 10 | Minimum number of billing minutes that will be used in the revenue calculations |
| *billingAuthorizationCode* | | | Billing authorization code |
| *billingProcessingCode* | | | Billing processing code |
| *payable* | Option (yes, no) | | Flag indicating wheither project is payable or not (ie. whether the project is included in the client's timekeeping report or not) |
| *payType* | Option (hour, visit) | | Pay type indicating whether project staff is paid per visit or per hour |
| *payRate* | Decimal | (9, 2) | Staff person pay rate per visit or per hour (depending on value of payType) |
| *payMinimumMinutes* | Integer | | Minimum number of pay minutes that will be used in the revenue calculations |
| *phoneSurveyFlag* | Option (true, false) | | Flag indicating if the phone survey is enabled for the project |
| *webSurveyFlag* | Option (true, false) | | Flag indicating if the web survey is enabled for the project |
| *checkOutFlag* | Option (true, false) | | Flag indicating if check-out is enabled for the project |
| *checkInFlag* | Option (true, false) | | Flag indicating if check-in is enabled for the project |
| *checkOutTimekeepingFlag* | Option (true, false) | | Flag indicating if check-in/check-out with timekeeping is enabled for the project |
| *ivrPhoneNumber* | Char | 25 | Telephone number if the IVR survey system is being used |
| *ivrVoiceSiteId* | Integer | | The voice site id for the phone survey |
| *ivrPageNumber* | Integer | | The page number on the voice site id for the phone survey |
| *faxRequiredFlag* | Option (true, false) | | Flag indicating if a fax is required for the project |

**Element: <project>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *faxNumber* | Char | 25 | The phone number for the fax system for this project |
| *faxCertificationStatement* | Char | 500 | The sentence included on all faxes for the certification that the project has been completed |
| *tagIdList* | Comma-separated list of tag ID values | | List of tags associated with the project |

# Packager Visit Data Descriptions

Natural Insight Packager can export XML visit[1] data including dates and times, management and other information for visits occurring in active projects according to the following search criteria:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <visit> contains all of the visit data attributes. Note that the elements <location> and <staff> are sub-elements of <visit>:

```
<visit>
   <location/>
   <staff/>
</visit>
```

<location> provides information on the location where the visit is located. <staff> provides information on the staff person assigned to the visit.

***EXAMPLE OF THE <VISIT> ELEMENT***

```
<visit id="275193" projectNumber="6" bundledScheduling="No" cre-
ateDate="2014-03-13 12:57:55.643"
updateDate="2014-03-28 15:16:23.077" resourceNumber="1"
managerRoleId="1" managerRoleTitle="Default Manager"
locationManagerId="4000" visitExpectedTime="120"
earliestStartDate="03/28/2014" latestStartDate="03/28/2014" earli-
estStartTime="" latestStartTime=""
daysOfWeek="Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday"
dataRequiredFlag="true" faxRequiredFlag="false"
currentScheduleStartDateTime="2014-03-28 15:00:00.000"
currentScheduleEndDateTime="2014-03-28 16:00:00.000"
previousScheduleStartDateTime="2014-03-28 13:00:00.000"
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

```
previousScheduleEndDateTime="2014-03-28 15:00:00.000"
dataReceivedFlag="true"
dataReceivedDate="2014-03-28 15:16:23.077"
faxReceivedFlag="false" faxReceivedDate=""
visitType="Imported with Assignment" extraEffortType="" extraEf-
fortAmount="" visitNote1="" managerNote1Flag=""
visitNote1CreateDate="" visitNote1UpdateDate="" visitNote2="" man-
agerNote2Flag="" visitNote2CreateDate=""
visitNote2UpdateDate="" visitNote3="" managerNote3Flag=""
visitNote3CreateDate="" visitNote3UpdateDate="">

   <location id="ST100" number="100"
   title="NATURAL INSIGHT STORE #ST100"
   locationGroupTitle3="Base Parent Location"
   locationGroupTitle2="Base Parent Location"
   locationGroupTitle1="Base Parent Location"
   address="3526 KING STREET" city="ALEXANDRIA" state="VA"
   postalCode="22302"/>

   <staff staffNumber="5003" firstName="Ann"
   lastName="Appleman" managerLevel2LastName="Robert, Rachel" man-
   agerLevel1LastName="Dash, Dan" regionCode=""/>

</visit>
```

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

**Element: <visit>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *Id* | Integer | | ID that uniquely identifies the visit |
| *projectNumber* | Integer | | Project ID that uniquely identifies a project |
| *bundledScheduling* | Option (Yes, No) | | Flag indicating whether the visit is part of a Bundled Pro-jector not |
| *createDate* | Date | | Date the visit was created |
| *updateDate* | Date | | Date something in the visit was last updated |
| *resourceNumber* | Integer | | Number of resources needed for each visit |
| *managerRoleId* | Integer | | Identifying number for the manager role ID of the visit |
| *managerRoleTitle* | Char | 250 | Title of the manager role for the visit |

**Element: <visit>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *locationManagerId* | | | |
| *earliestStartDate* | Date | | Earliest start date the visit can be scheduled |
| *latestStartDate* | Date | | Latest start date the visit can be scheduled |
| *earliestStartTime* | Time | | Earliest start time the visit can be scheduled |
| *latestStartTime* | Time | | Latest start time the visit can be scheduled |
| *daysOfWeek* | Option (Monday, Tuesday, Wednesday, Thursday, Fri-day, Saturday, Sunday) | | The days of the week when the visit can be scheduled (all options that apply are indicated) |
| *dataRequiredFlag* | Option (true, false) | | Flag indicating whether web survey data is required for the visit or not |
| *faxRequiredFlag* | Option (true, false) | | Flag indicating if a fax survey is required for the visit or not |
| *currentScheduleStartDateTime* | Date/Time | | The currently scheduled start date and time of the visit |
| *currentScheduleEndDateTime* | Date/Time | | The currently scheduled end date and time of the visit |
| *previousScheduleStartDateTime* | Date/Time | | The previously scheduled start date and time of the visit |
| *previousScheduleEndDateTime* | Date/Time | | The previously scheduled end date and time of the visit |
| *dataReceivedFlag* | Option (true, false) | | Flag indicating if web survey data has been received for the visit |
| *dataReceivedDate* | Date | | Date the web survey data was received |
| *faxReceivedFlag* | Option (true, false) | | Flag indicating if a fax survey has been received for the visit |
| *faxReceivedDate* | Date | | Date the fax was received |

**Element: <visit>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *visitType* | Option (single, flex, import, importAssign, plan, unknown) | | The type of visit indicating how the visit was created |
| *extraEffortType* | Option (”Percent of Pay” or “Stipend (Flat Rate)”) | | The type of extra effort pay that will be provided for this visit. Possible values are “Percent of Pay” or “Stipend (Flat Rate)”. |
| *extraEffortAmount* | Decimal | | The amount of extra effort pay the staff member who completes the visit will receive. If the value is “25.00”, for example, this indicates 25 dollars if “Stipend (Flat Rate)” is the value for the *extraEffortType* attribute. If “Percent of Pay” is the value of the *extraEffortType* attribute, “25.00” indicates 25 percent of the staff member’s total pay for the visit. Values for *extraEffortAmount* will be decimals with 2 places such as “25.00”. |
| *visitNote1* | Char | | The first visit note. |
| *managerNote1Flag* | Option (Yes, No) | | Flag indicating whether the first visit note will be visible to only managers - ie. *mgr1* or above level users - or to anyone with access to the visit. |
| *visitNote1CreateDate* | Date | | The creation date of the first visit note. |
| *visitNote1UpdateDate* | Date | | The date of the last update to the first visit note. |
| *visitNote2* | Char | | The second visit note. |
| *managerNote2Flag* | Option (Yes, No) | | Flag indicating whether the second visit note will be visible to only managers - ie. *mgr1* or above level users - or to anyone with access to the visit. |
| *visitNote2CreateDate* | Date | | The creation date of the second visit note. |
| *visitNote2UpdateDate* | Date | | The date of the last update to the second visit note. |
| *visitNote3* | Char | | The third visit note. |
| *managerNote3Flag* | Option (Yes, No) | | Flag indicating whether the third visit note will be visible to only managers - ie. *mgr1* or above level users - or to anyone with access to the visit. |

**Element: <visit>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *visitNote3CreateDate* | Date | | The creation date of the third visit note. |
| *visitNote3UpdateDate* | Date | | The date of the last update to the third visit note. |

# Packager Location Data Descriptions (Visit Level)

Natural InsightPackager can export XML location data (at the visit[1] level) - including location IDs, groupings and addresses - per visit. As such, <location> is a sub-element of <visit>. Visits in active projects that match the following search criteria will be returned in the XML packet:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <location> contains all of the location attributes.

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

**Sub-element of <visit>: <location>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *Id* | Char | 25 | Unique location ID of the location where the visit takes place |
| *Number* | Char | 50 | Number of the location where the visit takes place |
| *Title* | Char | 500 | Name of the location where the visit takes place |
| *locationGroupTitle3* | Char | 500 | |
| *locationGroupTitle2* | Char | 500 | |
| *locationGroupTitle1* | Char | 500 | |
| *Address* | Char | 500 | All street address information is to be contained in this attribute. There is no other street address attribute. |
| *City* | Char | 150 | |
| *State/Province* | Char | 2 | 2-letter code |

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

| Sub-element of <visit>: <location> | | | |
|---|---|---|---|
| XML Attribute | Data Type | Max Data Size | Description |
| *postalCode* | Char | 25 | 5 or 9 digit zip code |

# Packager Staff Data Descriptions (Visit Level)

Natural InsightPackager can export XML staff data - including staff ID and management - per visit[1]. As such, <staff> is a sub-element of <visit>. Visits in active projects that match the following search criteria will be returned in the XML packet:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <staff> contains all of the staff attributes.

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

**Sub-element of <visit>: <staff>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *staffNumber* | Char | 25 | Unique staff ID of the staff person assigned to the visit |
| *firstName* | Char | 100 | First name of the staff person assigned to the visit |
| *lastName* | Char | 100 | Last name of the staff person assigned to the visit |
| *managerLevel2LastName* | Char | 202 | Last name of the manager level 2 (mgr2) of the staff person assigned to the visit |
| *managerLevel1LastName* | Char | 202 | Last name of the manager level 1 (mgr1) of the staff person assigned to the visit |
| *regionCode* | Char | | |

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

# Packager Survey Question Data Descriptions (Visit Level)

Natural InsightPackager can export XML survey data (at the visit[1] level) describing the structure of the survey plus the survey's question content and possible answer choices for multiple choice questions. The surveys returned in the XML packet are associated with active projects and will fall into one of the following search criteria:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <survey> contains a <question> sub-element for each question in the survey. For any grid questions, the sub-element <gridQuestion> of the <question> element is used. Also, the sub-element <choice> of the <question> element indicates a multiple choice response value for either a radio button choice question (where one answer is accepted) or, alternatively, for a check box choice question (where one or more answers are accepted).

***EXAMPLE OF THE XML HIERARCHY OF THE <SURVEY> ELEMENT***

```
<survey type="Project-Based">
   <question>
      <choice/>
      <choice/>
   </question>
   <question></question>
   <question></question>
   <question>
      <gridQuestion></gridQuestion>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

```
            <gridQuestion>
               <choice/>
               <choice/>
            </gridQuestion>
            <gridQuestion>
               <choice/>
               <choice/>
               <choice/>
               <choice/>
            </gridQuestion>
         </question>
      </survey>
```

***EXAMPLE OF THE <SURVEY> ELEMENT AND THE <QUESTION> SUB-ELEMENT***

```
<survey id="13" type="Project-Based" title="Project Survey"
description="" startDate="2014-07-24 08:00:00.0"
endDate="2014-07-31 23:59:00.0"
availableUntil="2014-08-01 23:59:00.0">
   <question surveyItemId="56" variableName="nivar56" pageNumber="1"
   questionNumber="1" type="Radio Button" required="true" min=""
   max="" shortTitle="completion"
   text="Did you successfully complete this visit?">
      <choice displayValue="Yes" storeValue="Yes"/>
      <choice displayValue="No" storeValue="No"/>
   </question>
</survey>
```

Two types of surveys can be tied to a project:

1. a project-based survey <survey type="Project-Based">, and/or
2. a timekeeping survey <survey type="Timekeeping">.

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

## Survey Element

| Element: <survey> | | | |
|---|---|---|---|
| **XML Attribute** | **Data Type** | **Max Data Size** | **Description** |
| *Id* | Integer | | Unique ID that identifies the survey |
| *Type* | Option (Project-Based, Non-Project) | | Type of survey: project-based or not project-based |
| *Title* | Char | 150 | Title of the survey |
| *Description* | Char | 250 | Description of the survey |
| *startDate* | Date/Time | | The time and date when the survey is made available to staff |
| *endDate* | Date/Time | | The time and date before which the survey should be submitted |
| *availableUntil* | Date/Time | | The time and date after which the survey can no longer be accepted |

### Question Sub-Element

For each <survey>, there are <question> sub-elements.

| Sub-element of <survey>: <question> | | | | |
|---|---|---|---|---|
| **XML Sub-Element of <survey>** | **XML Attribute** | **Data Type** | **Max Data Size** | **Description** |
| <question> | *surveyItemId* | | | ID for the question within the survey |
| <question> | *variableName* | Char | 50 | Variable name of the question within the survey |
| <question> | *pageNumber* | Integer | | The page number on which the question is in the survey |
| <question> | *questionNumber* | Char | 15 | The number of the question within the survey |
| <question> | *Type* | Option (text, numeric, date only, time only, both date and time, currency, grid, photo, signature | | Type of question |

| Sub-element of <survey>: <question> | | | | |
|---|---|---|---|---|
| XML Sub-Element of <survey> | XML Attribute | Data Type | Max Data Size | Description |
| <question> | *Required* | Option (true, false) | | |
| <question> | *Min* | Char | 30 | The minimum value amount accepted as the answer to the question. For text questions, the amount is measured in text length. For numeric questions, the amount is measured as a number. For date questions, the amount is measured as a date. For time questions, the amount is measured as a time. |
| <question> | *Max* | Char | 30 | The maximum value amount accepted as the answer to the question. For text questions, the amount is measured in text length. For numeric questions, the amount is measured as a number. For date questions, the amount is measured as a date. For time questions, the amount is measured as a time. |
| <question> | *shortTitle* | Char | 50 | Short title describing the question |
| <question> | *Text* | Char | 3000 | The text of the question (can include HTML which will be rendered) |

## *Grid Question: Sub-Element of Question*

A grid question is a special type of survey question. It is a question at the intersection of a row and column - a cell - in a grid. <gridQuestion> is a sub-element of the <question> element .

***EXAMPLE OF A SURVEY WITH A GRID QUESTION***

This survey has one question - a grid question - <question type="grid">. The grid has one row and three columns equaling three cells or three <gridQuestion>s. Both the first and the third <gridQuestion> are multiple choice and thus contain <choice> tags indicating the different possible options.

```
<survey id="9" type="Project-Based" title="Daily Reset Ques-
tionnaire" description="Daily Reset Questionnaire"
startDate="2014-01-01 08:00:00.0" endDate="2014-12-31 23:59:00.0"
availableUntil="2015-01-03 23:59:00.0">
   <question surveyItemId="57" variableName="nivar57" pageNumber="5"
```

```
        questionNumber="5" type="Grid" required="true" min="" max=""
        shortTitle="product info" text="Tell me about the product:">

           <gridQuestion surveyItemId="58" variableName="nivar58"
           required="true" min="" max="" rowNum="1"
           rowTitle="Fizzy 12 oz. Bottle" colNum="1"
           columnTitle="Was the product in stock in the front?">

              <choice displayValue="Yes" storeValue="Yes"/>

              <choice displayValue="No" storeValue="No"/>

           </gridQuestion>

           <gridQuestion surveyItemId="59" variableName="nivar59"
           required="true" min="1" max="" rowNum="1"
           rowTitle="Fizzy 12 oz. Bottle" colNum="2"
           columnTitle="How many of the product were present?">

           </gridQuestion>

           <gridQuestion surveyItemId="60" variableName="nivar60"
           required="true" min="" max="" rowNum="1"
           rowTitle="Fizzy 12 oz. Bottle" colNum="3"
           columnTitle="What was the condition of the merchandise
           present?">

              <choice displayValue="All of the product items pristine /
              unopened / in order" storeValue="100%"/>

              <choice displayValue="75% of product items pristine /
              unopened / in order" storeValue="75%"/>

              <choice displayValue="50% of product items pristine /
              unopened / in order" storeValue="50%"/>

              <choice displayValue="25% of product items pristine /
              unopened / in order" storeValue="25%"/>

              <choice displayValue="All products items damaged / opened /
              in disarray" storeValue="0%"/>

           </gridQuestion>

        </question>

     </survey>
```

**Sub-element of <question>: <gridQuestion>**

| XML Sub-Element of <question> | XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|---|
| <gridQuestion> | *variableName* | Char | 50 | Variable name given to the cell (the row/column combination) of the grid question |

| Sub-element of <question>: <gridQuestion> | | | | |
|---|---|---|---|---|
| **XML Sub-Element of <question>** | **XML Attribute** | **Data Type** | **Max Data Size** | **Description** |
| <gridQuestion> | *Required* | Option (true, false) | 1 | Flag indicating whether the response to the grid question is required or not required (i.e. optional) |
| <gridQuestion> | *Min* | Char | 30 | The minimum value amount accepted as the answer to the question. For text questions, the amount is measured in text length. For numeric questions, the amount is measured as a number. For date questions, the amount is measured as a date. For time questions, the amount is measured as a time. |
| <gridQuestion> | *Max* | Char | 30 | The maximum value amount accepted as the answer to the question. For text questions, the amount is measured in text length. For numeric questions, the amount is measured as a number. For date questions, the amount is measured as a date. For time questions, the amount is measured as a time. |
| <gridQuestion> | *rowNum* | Integer | 10 | Row number of the question within the grid |
| <gridQuestion> | *rowTitle* | Char | 500 | Title of the row of the question within the grid |
| <gridQuestion> | *colNum* | Integer | 10 | Column number of the question within the grid |
| <gridQuestion> | *columnTitle* | Char | 500 | Title of the column of the question within the grid |

### *Multiple Choice: Sub-Element of Question or Grid Question*

A choice (multiple choice) question is a special type of survey <question> or <gridQuestion>. The value (response/answer) of the multiple choice question is indicated by the <choice>sub-element. <choice> is a sub-element of either the <question> element or the <gridQuestion> element.

| Sub-element of <question> or <gridQuestion>: <choice> | | | | |
|---|---|---|---|---|
| **XML Sub-Element of <question> or <gridQuestion>** | **XML Attribute** | **Data Type** | **Max Data Size** | **Description** |
| <choice> | *displayValue* | | | A possible answer choice (value) that the survey taker sees when completing the multiple choice question. |
| <choice> | *storeValue* | | | The value stored in the database that is associated with the displayValue the survey taker selects as the answer/response to the multiple choice <question> or <gridQuestion> |

# Packager Survey Response Data Descriptions (Visit Level)

Natural InsightPackager can export XML survey response data (at the visit[1] level) that includes the responses a staff person has provided for each of the questions in a survey. The survey response data returned in the XML packet is associated with active projects. The following are the three possible search criteria when requesting the XML packet:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <data> contains a <response> sub-element for each question's response in a survey. Since each survey question has a variable name, this name can be referenced in the <response> sub-element to link a survey answer to its corresponding question.

***EXAMPLE OF THE XML HIERARCHY OF THE <DATA> ELEMENT***

```
<data>
   <response/>
   <response/>
   <response/>
   <response/>
</data>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

***EXAMPLE OF THE <DATA> ELEMENT AND <RESPONSE> SUB-ELEMENT***

```
<data surveyId="13" responseDate="2014-07-24T09:52:41.450"
visitId="275285" responseId="167F3E88-007B-D621-FEB36A3ABE4FD625">

   <response var="nivar56" val="Yes"/>

</data>
```

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

## Data Element

| Element: <data> | | | |
|---|---|---|---|
| **XML Attribute** | **Data Type** | **Max Data Size** | **Description** |
| *surveyId* | Integer | | Unique ID that identifies the survey |
| *responseDate* | Date/Time | | The date and time the survey was submitted |
| *visitId* | Integer | | If applicable, the ID of the visit associated with the survey data (project-based only) |
| *responseId* | | | |

## Response Sub-Element

Each <response> sub-element of the <data> element represents each question's response in a survey. The <response> var attribute represents the variable name identifying the question while its val attribute represents the value of the variable. The value is either

a. the stored value (*storeValue* attribute of the <choice> tag from questions in a <survey>) corresponding to the response a staff person selects (*displayValue* attribute of the <choice> tag from questions in a <survey>) when answering a multiple choice question. or

b. the free response a staff person provides when answering question types other than multiple choice.

**Sub-element of <data>: <response>**

| XML Sub-Element of <data> | XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|---|
| <response> | *var* | | | The variable name identifying a question |
| <response> | *val* | | | The stored value of the response to the question |

# Packager Timekeeping Data Descriptions (Visit Level)

Natural InsightPackager can export XML timekeeping data per visit[1] for active projects. This timekeeping information can be used for payroll purposes and customized for direct integration with client payroll and billing systems. The following are the three possible search criteria when requesting the XML packet:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

The XML element <timekeepingData> contains a <response> tag nested within a <punch> tag for each punch in a timekeeping survey. Each punch has a type - work, travel, break or lunch - that is referenced in the type attribute of the <response> tag.

***EXAMPLE OF THE XML HIERARCHY OF THE <TIMEKEEPINGDATA> ELEMENT***

```
<timekeepingData>
   <punch>
      <response/>
   </punch>
   <punch>
      <response/>
   </punch>
</timekeepingData>
```

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

### EXAMPLE OF THE <TIMEKEEPINGDATA> ELEMENT WITH PUNCHES FOR WORK, BREAK AND LUNCH

```
<timekeepingData responseId="1000" surveyId="14" staffNumber="5003"
visitId="275298" sourceCode="web" mileage="0" travelMinutes="0"
lunchMinutes="0" createDate="2014-10-03 12:03:26.333"
updateDate="2014-10-03 12:03:26.333">

   <punch>

      <response type="Work" start="2014-10-03 10:00:00.000"
      end="2014-10-03 10:15:00.000"/>

   </punch>

   <punch>

      <response type="Break" start="2014-10-03 10:16:00.000"
      end="2014-10-03 10:30:00.000"/>

   </punch>

   <punch>

      <response type="Work" start="2014-10-03 10:31:00.000"
      end="2014-10-03 10:45:00.000"/>

   </punch>

   <punch>

      <response type="Lunch" start="2014-10-03 10:46:00.000"
      end="2014-10-03 11:00:00.000"/>

   </punch>

   <punch>

      <response type="Work" start="2014-10-03 11:01:00.000"
      end="2014-10-03 11:19:00.000"/>

   </punch>

</timekeepingData>
```

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

## Timekeeping Data Element

**Element: <timekeepingData>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *responseId* | Integer | | An internal Natural Insight ID that must be included in the XML. It is not a unique ID, so please do not use it for any reference purposes. |
| *surveyId* | Integer | | Unique ID that identifies the timekeeping survey |
| *staffNumber* | Char | 25 | Staff ID of the staff person associated with the timekeeping record |
| *visitId* | Integer | | The ID of the visit associated with the timekeeping record |
| *sourceCode* | Option (web, phone, check-in, unknown) | | How the timekeeping data was entered |
| *mileage* | Integer | | Amount of mileage reported |
| *travelMinutes* | Integer | | Number of travel minutes reported |
| *lunchMinutes** | Integer | | Number of minutes reported for a lunch break |
| *createDate* | Date/Time | | The date and time the timekeeping record was created |
| *updateDate* | Date/Time | | The date and time the timekeeping record was last updated |

* Client-specific timekeeping questions will be included as attributes after the *lunchMinutes* attribute in the <timekeeping> element. Contact Natural Insight Client Services to include customized questions or data fields to your timekeeping survey.

## Response Sub-Element

Each <response> sub-element is wrapped within an open and close <punch> tag, a sub-element of the <timekeepingData> element. Each <response> tag represents a staff person's timekeeping punch for work, travel, break or lunch in a timekeeping survey.

The hierachy is as follows:

```
    <timekeepingData><punch><response/></punch></timekeepingData>.
```

**Sub-element of <punch>: <response>**

| XML Sub-Element of <punch> | XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|---|
| <response> | *type* | Option (work, lunch, travel, break) | | Type of punch recorded (work, lunch, travel or break) |
| <response> | *start* | Date/Time | | The start date and time of the punch |
| <response> | *end* | Date/Time | | The end date and time of the punch |

# Packager Web Service Methods

The Natural InsightPackager SOAP-based web service allows clients to export Natural Insight data as a data package according to one of the three following search criteria:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

Because the processing of the data package can potentially take a long time to complete (because the data package may be very large - possibly many megabytes), the retrieval process is broken down into three steps, each with a corresponding public method:

1. **requesting the package (*requestPackage* method - Step 1),**
2. **checking the status of the package build (*getStatus* method - Step 2),** and
3. **retrieving the package when it is complete (*getPackage* method - Step 3).**

The three public methods in the Natural Insight Packager web service and can be consumed using either the SOAP protocol over HTTPS or simple HTTP GET/POST requests. The WSDL definition for the Natural Insight Packager web service, located at the URL, https://my.naturalinsight.com/packagerWebService.cfc?wsdl, describes its methods and arguments in detail.

For simple access to the Natural Insight Packager web service, and to better understand the steps neces-sary to successfully generate a data package, the following URLs (in which the bold values are replaced as needed) can be used to invoke the three different methods.

## Step 1: The requestPackage Method

There are three versions of the *requestPackage* method that each refer to the different search criterion used:

1. date range (*requestPackageByDate* method),
2. project ID (*requestPackageByProjectNumber* method), or
3. survey ID and date range (*requestPackageBySurveyAndDate* method).

Whichever *requestPackage* method is called, it will return the UUID of the data package. This UUID should be stored locally and is required in the remaining two methods - *getStatus* and *getPackage*.

### requestPackage Method by Date Range

To request a package by date range, use the following URL:

https://my.naturalinsight.com/
packagerWebService.cfc?wsdl&method=requestPackageByDate&authCd=123-456&
startDate=2014-01-01 00:00:00&endDate=2014-01-02 00:00:00

### requestPackage Method by Project ID

To request a package by project ID, use the following URL:

https://my.naturalinsight.com/
packagerWebService.cfc?wsdl&method=requestPackageByProjectNumber&authCd=123-456&
projectNumber=1234

### requestPackage Method by Survey ID and Date Range

To request a package by survey ID and date range, use the following URL:

https://my.naturalinsight.com/
packagerWebService.cfc?wsdl&method=requestPackageBySurveyIdAndDate&authCd=123-456&
surveyId=1234&startDate=2014-01-01 00:00:00&endDate=2014-12-12 00:00:00

## Step 2: The getStatus Method

To check the data package status, call the *getStatus* method at the following URL:

https://my.naturalinsight.com/packagerWebService.cfc?wsdl&method=getStatus&
authCd=123-456&packageUUID=2BD9EC0D-0649-13E7-1D35344478D1A8CF

During the package processing, the status will return as either:

- "Processing" along with the percentage complete, or
- "Complete" once the package processing is done.

Once the package processing is complete, it can now be retrieved using the *getPackage* method.

## Step 3: The getPackage Method

To retrieve the data package, call the *getPackage* method at the following URL:

https://my.naturalinsight.com/packagerWebService.cfc?wsdl&method=getPackage&
authCd=123-456&packageUUID=30247061-B421-86F4-AC9A2FF7F7BFA34B

The *getPackage* method returns the data package in the default format of WDDX (Web Distributed Data eXchange) - a string of XML-encoded data.

## Return Formats of the getPackage Method

Though the default return format of the *getPackage* method is WDDX, you can also request the package as a string only or in JSON (JavaScript Object Notation). To do so, append "&returnformat=JSON" or "&returnformat=plain" at the end of the *getPackage* URL to indicate the preferred return format. Thus, the possible parameter values for *returnformat* are:

1. WDDX (the default)
2. Plain (for a string only)
3. JSON

## Availability of the Data Package from the getPackage Method

Data packages will be available for one week after creation. After a week, the package will be deleted from the system. The package represents a snapshot of the data at the date and time the package was created and is not updated with subsequent changes.

# Packager Web Service API Summary

## URL

https://my.naturalinsight.com/packagerWebService.cfc

## WSDL

https://my.naturalinsight.com/packagerWebService.cfc?wsdl

## Methods

### requestPackage Methods

#### *requestPackageByDate Method*

```
<wsdl:operation name="requestPackageByDate" parameterOrder="authCd
startDate endDate">
   <wsdl:input message="impl:requestPackageByDateRequest"
   name="requestPackageByDateRequest"/>
   <wsdl:output message="impl:requestPackageByDateResponse"
   name="requestPackageByDateResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

#### *requestPackageByDate Parameters*

1. required string *authCd*
2. required *startDate* in ISO 8601 format
3. required *endDate* in ISO 8061 format

#### *requestPackageByProjectNumber Method*

```
<wsdl:operation name="requestPackageByProjectNumber"
parameterOrder=" authCd projectNumber">
   <wsdl:input message="impl:requestPackageByProjectNumberRequest" name-
   e="requestPackageByProjectNumberRequest"/>
```

```
        <wsdl:output message="impl:requestPackageByProjectNumberResponse"
        name="requestPackageByProjectNumberResponse"/>

        <wsdl:fault message="impl:CFCInvocationException"
        name="CFCInvocationException"/>

    </wsdl:operation>
```

### *requestPackageByProjectNumber Parameters*

1. required string *authCd*
2. required integer *projectNumber* (project ID)

### *requestPackageBySurveyIdAndDate Method*

```
    <wsdl:operation name="requestPackageBySurveyIdAndDate" parameterOrder="authCd
    surveyId startDate endDate">

        <wsdl:input message="impl:requestPackageBySurveyIdAndDateRequest"
        name="requestPackageBySurveyIdAndDateRequest"/>

        <wsdl:output message="impl:requestPackageBySurveyIdAndDateResponse"
        name="requestPackageBySurveyIdAndDateResponse"/>

        <wsdl:fault message="impl:CFCInvocationException"
        name="CFCInvocationException"/>

    </wsdl:operation>
```

### *requestPackageBySurveyIdAndDate Parameters*

1. required string *authCd*
2. required integer *surveyId*
3. required *startDate* in ISO 8601 format[1]
4. required *endDate* in ISO 8061 format

---

[1]*The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values.*

*The time must be in the ISO 8061 format - hh:mm or hh:mm:ss - with hh, mm and ss being 2 characters, padded in front with a 0 for single-digit values.*

*The date precedes the time with the resulting combined date/time format as "2014-04-05 14:30:00".*

## requestPackage Description

The *requestPackage* methods - *requestPackageByDate*, *requestPackageByProjectNumber* or *requestPackageBySurveyIdAndDate* - allow you to request a data package either by date range, by project ID or by survey ID and date range respectively and returns the UUID of the package.

### *requestPackage Potential Error Messages*

## getStatus Method

```
<wsdl:operation name="getStatus" parameterOrder="authCd packageUUID">

   <wsdl:input message="impl:getStatusRequest" name="getStatusRequest"/>

   <wsdl:output message="impl:getStatusResponse" name="getStatusResponse"/>

   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>

</wsdl:operation>
```

### *getStatus Parameters*

1. required string *authCd*
2. required string *packageUUID*

### *getStatus Description*

The Natural Insight Packager *getStatus* method checks on the status of the data package retrieval and will return "Processing" with the percentage complete if the package is still processing, or "Complete" if the package processing is complete.

## getPackage Method

```
<wsdl:operation name="getPackage" parameterOrder="authCd packageUUID">

   <wsdl:input message="impl:getPackageRequest" name="getPackageRequest"/>

   <wsdl:output message="impl:getPackageResponse" name="getPackageResponse"/>

   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>

</wsdl:operation>
```

## *getPackage Parameters*

1. required string *authCd*
2. required string *packageUUID*
3. optional string *returnFormat* [Plain, JSON or WDDX (the default)]

## *getPackage Description*

The Natural Insight Packager *getPackage* returns the data package in WDDX (the default), JSON or as a string only (Plain).

Chapter 7

# Natural Insight Packager Deleted Visits Web Service

# Packager - Deleted Visits - Web Service Overview

The Natural Insight Packager - Deleted Visits - web service exports deleted visits[1] for your client instance of Natural Insight.

## Description

Retrieve and use deleted visit data by date range.

***EXAMPLE OF XML HIERARCHY RETURNED WHEN THE XML PACKAGER - DELETED VISITS-DATA IS REQUESTED***

```
<?xml version="1.0" encoding="utf-8"?>
<root>
   <project/>
   <project/>
      <visitDeleted/>
      <visitDeleted/>
      <visitDeleted/>
</root>
```

Each XML packet of the Natural Insight Packager - Deleted Visits - web service is organized into project information and deleted visit information. (Deleted visit attributes contain location information about where the visit was performed and staff information about the staff person assigned to the visit.)

[1]*Whenever "a visit" is used in this text, understand that "a visit" is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

Refer to the data description tables for each category in the following bulleted list to discover each element and its attributes.

- Project data (visit level)
- Deleted visit data
    - Location data (visit level) as attributes
    - Staff data (visit level) as attributes

### Frequency

On-demand

## WSDL

You can access the WSDL for the Natural Insight Packager API at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://niWeb" xmlns:intf="http://niWeb"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://rpc.xml.coldfusion"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://niWeb">

<!-- WSDL created by ColdFusion version 9,0,1,274733 -->

<wsdl:types>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://rpc.xml.coldfusion">

<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType name="CFCInvocationException">

<sequence/>

</complexType>

</schema>

</wsdl:types>

<wsdl:message name="requestPackageResponse">

<wsdl:part name="requestPackageReturn" type="xsd:string"/>

</wsdl:message>

<wsdl:message name="requestPackageBySurveyIdAndDateResponse">

<wsdl:part name="requestPackageBySurveyIdAndDateReturn" type="xsd:string"/>
```

```
</wsdl:message>
<wsdl:message name="requestPackageByProjectNumberRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="projectNumber" type="xsd:double"/>
</wsdl:message>
<wsdl:message name="requestPackageDeletedVisitByDateResponse">
<wsdl:part name="requestPackageDeletedVisitByDateReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="CFCInvocationException">
<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="requestPackageRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="projectNumber" type="xsd:double"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByProjectNumberResponse">
<wsdl:part name="requestPackageByProjectNumberReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageBySurveyIdAndDateRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="surveyId" type="xsd:double"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getStatusRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="packageUUID" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getStatusResponse">
<wsdl:part name="getStatusReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getPackageResponse">
<wsdl:part name="getPackageReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageDeletedVisitByDateRequest">
```

```
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getPackageRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="packageUUID" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByDateRequest">
<wsdl:part name="authCd" type="xsd:string"/>
<wsdl:part name="startDate" type="xsd:string"/>
<wsdl:part name="endDate" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="requestPackageByDateResponse">
<wsdl:part name="requestPackageByDateReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="packagerWebService">
<wsdl:operation name="getPackage" parameterOrder="authCd packageUUID">
<wsdl:input message="impl:getPackageRequest" name="getPackageRequest"/>
<wsdl:output message="impl:getPackageResponse" name="getPackageResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
<wsdl:operation name="getStatus" parameterOrder="authCd packageUUID">
<wsdl:input message="impl:getStatusRequest" name="getStatusRequest"/>
<wsdl:output message="impl:getStatusResponse" name="getStatusResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
<wsdl:operation name="requestPackageByDate" parameterOrder="authCd startDate
endDate">
<wsdl:input message="impl:requestPackageByDateRequest"
name="requestPackageByDateRequest"/>
<wsdl:output message="impl:requestPackageByDateResponse"
name="requestPackageByDateResponse"/>
<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>
</wsdl:operation>
```

```
<wsdl:operation name="requestPackageBySurveyIdAndDate"
parameterOrder="authCd surveyId startDate endDate">

<wsdl:input message="impl:requestPackageBySurveyIdAndDateRequest"
name="requestPackageBySurveyIdAndDateRequest"/>

<wsdl:output message="impl:requestPackageBySurveyIdAndDateResponse"
name="requestPackageBySurveyIdAndDateResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackageByProjectNumber" parameterOrder="authCd
projectNumber">

<wsdl:input message="impl:requestPackageByProjectNumberRequest"
name="requestPackageByProjectNumberRequest"/>

<wsdl:output message="impl:requestPackageByProjectNumberResponse"
name="requestPackageByProjectNumberResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackage" parameterOrder="authCd projectNumber
startDate endDate">

<wsdl:input message="impl:requestPackageRequest"
name="requestPackageRequest"/>

<wsdl:output message="impl:requestPackageResponse"
name="requestPackageResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

<wsdl:operation name="requestPackageDeletedVisitByDate"
parameterOrder="authCd startDate endDate">

<wsdl:input message="impl:requestPackageDeletedVisitByDateRequest"
name="requestPackageDeletedVisitByDateRequest"/>

<wsdl:output message="impl:requestPackageDeletedVisitByDateResponse"
name="requestPackageDeletedVisitByDateResponse"/>

<wsdl:fault message="impl:CFCInvocationException"
name="CFCInvocationException"/>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="packagerWebService.cfcSoapBinding"
type="impl:packagerWebService">

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="getPackage">

<wsdlsoap:operation soapAction=""/>
```

```
<wsdl:input name="getPackageRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getPackageResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="getStatus">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="getStatusRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="getStatusResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageByDate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageByDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageByDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">
```

```
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageBySurveyIdAndDate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageBySurveyIdAndDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageBySurveyIdAndDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageByProjectNumber">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageByProjectNumberRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageByProjectNumberResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackage">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>
```

```
</wsdl:input>

<wsdl:output name="requestPackageResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

<wsdl:operation name="requestPackageDeletedVisitByDate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="requestPackageDeletedVisitByDateRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:input>

<wsdl:output name="requestPackageDeletedVisitByDateResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://niWeb" use="encoded"/>

</wsdl:output>

<wsdl:fault name="CFCInvocationException">

<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://niWeb" use="encoded"/>

</wsdl:fault>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="packagerWebServiceService">

<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Natural
Insight packager Web service component</wsdl:documentation>

<wsdl:port binding="impl:packagerWebService.cfcSoapBinding"
name="packagerWebService.cfc">

<wsdlsoap:address
location="http://my.naturalinsight.com/packagerWebService.cfc"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

# Packager - Deleted Visits - Project Data Descriptions (Visit Level)

Natural Insight Packager - Deleted Visits - can export project data at the visit[1] level (including project dates, management, billing and contact information) for active projects with deleted visits according to date range.

The XML element <project> contains all of the project data attributes.

***EXAMPLE OF THE <PROJECT> ELEMENT***

```
<project number="6" bundledScheduling=”Yes”
createDate="2014-03-13 12:05:22.857"
updateDate="2014-09-15 11:49:52.967" title="Daily Reset"
shortTitle="Daily Reset" shortDescription="Daily reset merch"
fullDescription="Reset clothing and accessory merchandise 1x/day"
organizationName="Calvin Klein" organizationCode="277"
startDate="01/01/2014" endDate="12/31/2014"
expectedTimeMin="60" expectedTimeMax="60"
projectScheduleTime="60" projectManagerStaffNumber="0001"
projectManagerFirstName="Gordon" projectManagerLastName="Go" status-
s="Active" continuousCoverageFlag="false"
billingType="hour" billingRate="0.00"
billingMinimumMinutes="0" billingAuthorizationCode=""
billingProcessingCode="" payable="No" payType="visit"
payRate="20.00" payMinimumMinutes="0" phoneSurveyFlag="false"
webSurveyFlag="true" checkOutFlag="true" checkInFlag="false"
checkOutTimekeepingFlag="false" ivrPhoneNumber="" ivrVoiceSiteId="0"
ivrPageNumber="1" faxRequiredFlag="false" faxNumber="" faxCer-
tificationStatement="" surveyId="9" tagIdList=""/>
```

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

[1]*Whenever “a visit” is used in this text, understand that “a visit” is one of many terms Natural Insight clients may use to refer to work being done in a particular location at a particular time on a particular date. Some clients refer to visits as calls, visits, assignments, or activations as just a few examples. In the retail world, clients frequently use the term task or to-do.*

**Element: <project>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *Number* | Integer | | Project ID that uniquely identifies a project |
| *bundledScheduling* | Option (Yes, No) | | Flag indicating whether the project is a Bundled Projector not |
| *createDate* | Date | | Date the project was created |
| *updateDate* | Date | | Date when any information on the project was last updated |
| *Title* | Char | 500 | Title of the project |
| *shortTitle* | Char | 150 | Short title of the project |
| *shortDescription* | Char | 150 | Short description of the project |
| *fullDescription* | Char | 3000 | Full description of the project |
| *organizationName* | Char | 50 | Name of the organization (client) tied to the project |
| *organizationCode* | Char | 150 | Organization code |
| *startDate* | Date | | Start date of project |
| *endDate* | Date | | End date of project |
| *expectedTimeMin* | Integer | | Minimum number of minutes expected to complete |
| *expectedTimeMax* | Integer | | Maximum number of minutes expected to complete |
| *projectManagerStaffNumber* | Char | 25 | Project manager's unique staff ID |
| *projectManagerFirstName* | Char | 100 | Project manager's first name |
| *projectManagerLastName* | Char | 100 | Project manager's last name |
| *Status* | Option (active, inactive, pending, deleted, unknown) | | Status of the project |
| *continuousCoverageFlag* | Option (true, false) | | Flag indicating continuous coverage |
| *billingType* | Option (hour, visit) | | Billing type indicating whether this project is billed per visit or per hour |
| *billingRate* | Decimal | (9, 2) | Billing rate per visit or per hour (depending on the value of billingType) |

**Element: <project>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *billingMinimumMinutes* | Integer | 10 | Minimum number of billing minutes that will be used in the revenue calculations |
| *billingAuthorizationCode* | | | Billing authorization code |
| *billingProcessingCode* | | | Billing processing code |
| *payable* | Option (yes, no) | | Flag indicating wheither project is payable or not (ie. whether the project is included in the client's timekeeping report or not) |
| *payType* | Option (hour, visit) | | Pay type indicating whether project staff is paid per visit or per hour |
| *payRate* | Decimal | (9, 2) | Staff person pay rate per visit or per hour (depend-ing on value of payType) |
| *payMinimumMinutes* | Integer | | Minimum number of pay minutes that will be used in the revenue calculations |
| *phoneSurveyFlag* | Option (true, false) | | Flag indicating if the phone survey is enabled for the project |
| *webSurveyFlag* | Option (true, false) | | Flag indicating if the web survey is enabled for the project |
| *checkOutFlag* | Option (true, false) | | Flag indicating if check-out is enabled for the pro-ject |
| *checkInFlag* | Option (true, false) | | Flag indicating if check-in is enabled for the project |
| *checkOutTimekeepingFlag* | Option (true, false) | | Flag indicating if check-in/check-out with time-keeping is enabled for the project |
| *ivrPhoneNumber* | Char | 25 | Telephone number if the IVR survey system is being used |
| *ivrVoiceSiteId* | Integer | | The voice site id for the phone survey |
| *ivrPageNumber* | Integer | | The page number on the voice site id for the phone survey |
| *faxRequiredFlag* | Option (true, false) | | Flag indicating if a fax is required for the project |
| *faxNumber* | Char | 25 | The phone number for the fax system for this pro-ject |
| *faxCertificationStatement* | Char | 500 | The sentence included on all faxes for the cer-tification that the project has been completed |

**Element: <project>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *tagIdList* | Comma-separated list of tag ID values | | List of tags associated with the project |

# Packager - Deleted Visits - Visit Deleted Data Descriptions

Natural Insight Packager - Deleted Visits - can export XML deleted visit data including dates and times, location, staff and other information for visits occurring in active projects for a given date range.

The XML element <visitDeleted> contains all of the deleted visit data attributes.

***EXAMPLE OF THE <VISITDELETED> ELEMENT***

```
<visitDeleted id="275048" projectNumber="1" locationCode="SR1899"
deletedDate="2014-01-29 07:57:52.730" dataFlag="false" faxFlag-
g="false" assignToStaffNumber="5000" assignToName="Lagarde, Lila"
deleteByStaffNumber="7432" deleteByName="Grisby, John"/>
```

Access the Natural Insight Packager XML-based WSDL at https://my.naturalinsight.com/packagerWebService.cfc?wsdl.

**Element: <visitDeleted>**

| XML Attribute | Data Type | Max Data Size | Description |
|---|---|---|---|
| *Id* | Integer | | ID that uniquely identifies the deleted visit |
| *projectNumber* | Integer | | Project ID that uniquely identifies a project |
| *locationCode* | Char | 100 | Location ID of the location where the deleted visit is tied |
| *deletedDate* | Date | | The date the visit was deleted |
| *dataFlag* | Option (true, false) | 1 | Flag indicating if the survey data associated with the visit was deleted when the visit was deleted |
| *faxFlag* | Option (true, false) | 1 | Flag indicating if the fax associated with the visit was deleted when the visit was deleted |
| *assignToStaffNumber* | Char | 25 | The Staff ID of the staff person who was assigned to the deleted visit |
| *assignToName* | Char | 202 | The “lastname, firstname” of the staff person to whom the deleted visit was assigned |

| Element: <visitDeleted> | | | |
|---|---|---|---|
| XML Attribute | Data Type | Max Data Size | Description |
| *deleteByStaffNumber* | Char | 25 | The Staff ID of the staff person who deleted the visit |
| *deleteByName* | Char | 202 | The “lastname, firstname” of the person who deleted the visit |

# Packager - Deleted Visits - Web Service Methods

The Natural Insight Packager - Deleted Visits - SOAP-based web service allows clients to export Natural Insight deleted visit data as a data package according to date range.

Because the processing of the data package can potentially take a long time to complete (because the data package may be very large - possibly many megabytes), the retrieval process is broken down into three steps, each with a corresponding public method:

1. **requesting the package by date range (*requestPackageDeletedVisitsByDate* method - Step 1),**
2. **checking the status of the package build (*getStatus* method - Step 2),** and
3. **retrieving the package when it is complete (*getPackage* method - Step 3).**

The three public methods in the Natural Insight Packager - Deleted Visits- web service and can be consumed using either the SOAP protocol over HTTPS or simple HTTP GET/POST requests. The WSDL definition for the Natural Insight Packager web service, located at the URL, https://my.naturalinsight.com/packagerWebService.cfc?wsdl, describes its methods and arguments in detail.

For simple access to the Natural Insight Packager - Deleted Visits - web service, and to better understand the steps necessary to successfully generate a data package, the following URLs (in which the bold values are replaced as needed) can be used to invoke the three different methods.

## Step 1: The requestPackageDeletedVisitByDate Method

The method will return the UUID of the data package. This UUID should be stored locally and is required in the remaining two methods - *getStatus* and *getPackage*. To request the package by date range, use the following URL:

https://my.naturalinsight.com/
packagerWebService.cfc?wsdl&method=requestPackageDeletedVisitByDate&
authCd=123-456&startDate=2014-01-01 00:00:00&endDate=2014-01-02 00:00:00

## Step 2: The getStatus Method

To check the data package status, use the following URL:

https://my.naturalinsight.com/packagerWebService.cfc?wsdl&method=getStatus&
authCd=123-456&packageUUID=2BD9EC0D-0649-13E7-1D35344478D1A8CF

During the package processing, the status will return as either:

- "Processing" along with the percentage complete, or
- "Complete" once the package processing is done.

Once the package processing is complete, it can now be retrieved using the *getPackage* method.

## Step 3: The getPackage Method

To retrieve the data package, use the following URL:

https://my.naturalinsight.com/packagerWebService.cfc?wsdl&method=getPackage&
authCd=123-456&packageUUID=30247061-B421-86F4-AC9A2FF7F7BFA34B

The *getPackage* method returns the data package in the default format of WDDX (Web Distributed Data eXchange) - a string of XML-encoded data.

### Return Formats of the getPackage Method

Though the default return format of the *getPackage* method is WDDX, you can also request the package as a string only or in JSON (JavaScript Object Notation). To do so, append "&returnformat=JSON" or "&re-turnformat=plain" at the end of the *getPackage* URL to indicate the preferred return format. Thus, the possible parameter values for *returnformat* are:

1. WDDX (the default)
2. Plain (for a string only)
3. JSON

### Availability of the Data Package from the getPackage Method

Data packages will be available for one week after creation. After a week, the package will be deleted from the system. The package represents a snapshot of the data at the date and time the package was created and is not updated with subsequent changes.

# Packager - Deleted Visits - Web Service API Summary

## URL

https://my.naturalinsight.com/packagerWebService.cfc

## WSDL

https://my.naturalinsight.com/packagerWebService.cfc?wsdl

## Methods

### requestPackageDeletedVisitByDate Method

```
<wsdl:operation name="requestPackageDeletedVisitByDate"
parameterOrder="authCd startDate endDate">
   <wsdl:input message="impl:requestPackageDeletedVisitByDateRequest"
   name="requestPackageDeletedVisitByDateRequest"/>
   <wsdl:output message="impl:requestPackageDeletedVisitByDateResponse"
   name="requestPackageDeletedVisitByDateResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

#### *requestPackageDeletedVisitByDate Parameters*

1. required string *authCd*
2. required *startDate* in ISO 8601 format[1]
3. required *endDate* in ISO 8601 format

[1]*The date must be in the ISO 8061 format - yyyymmdd or yyyy-mm-dd - with mm and dd being 2 characters, padded in front with a 0 for single-digit values.*

*The time must be in the ISO 8061 format - hh:mm or hh:mm:ss - with hh, mm and ss being 2 characters, padded in front with a 0 for single-digit values.*

*The date precedes the time with the resulting combined date/time format as "2014-04-05 14:30:00".*

### *requestPackageDeletedVisitByDate Description*

The *requestPackage* methods - *requestPackageByDate*, *requestPackageByProjectNumber* or *requestPackageBySurveyIdAndDate* - allow you to request a data package either by date range, by project ID or by survey ID and date range respectively and returns the UUID of the package.

## getStatus Method

```
<wsdl:operation name="getStatus" parameterOrder="authCd packageUUID">
   <wsdl:input message="impl:getStatusRequest" name="getStatusRequest"/>
   <wsdl:output message="impl:getStatusResponse" name="getStatusResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

### *getStatus Parameters*

1. required string *authCd*
2. required string *packageUUID*

### *getStatus Description*

The Natural Insight Packager *getStatus* method checks on the status of the data package retrieval and will return “Processing” with the percentage complete if the package is still processing, or “Complete” if the package processing is complete.

## getPackage Method

```
<wsdl:operation name="getPackage" parameterOrder="authCd packageUUID">
   <wsdl:input message="impl:getPackageRequest" name="getPackageRequest"/>
   <wsdl:output message="impl:getPackageResponse" name="getPackageResponse"/>
   <wsdl:fault message="impl:CFCInvocationException"
   name="CFCInvocationException"/>
</wsdl:operation>
```

### *getPackage Parameters*

1. required string *authCd*
2. required string *packageUUID*

3. optional string *returnFormat* [Plain, JSON or WDDX (the default)]

### getPackage Description

The Natural Insight Packager *getPackage* returns the data package in WDDX (the default), JSON or as a string only (Plain).

Chapter 8

# Staff Maintenance Export Methods

# getStaff and getStaffMaintReport Export Methods (of Staff Maintenance) Overview

## getStaff Export Method of Staff Maintenance

The *getStaff* method is a part of the Staff Maintenance web service and can be called to export information about a single staff member within Natural Insight at a time. See **The getStaff Method** on page 54 for more information.

## getStaffMaintReport Export Method of Staff Maintenance

The *getStaffMaintReport* method is also part of the Staff Maintenance web service and can be called to export information about staff member operations - adds, updates and terminations - within Natural Insight for a given date range. See **The getStaffMaintReport Method** on page 56 for more information.