SafetyNet Web Services

Application Program Interface (API) Reference Document

November 11, 2017

Table of Contents

Table of Figures

Revision History

Revision Date	Who	Description
Oct 03, 2013	Edward P. Leisman	Initial Document
Oct 03, 2013	Joshua W. Burns	Reworked chapter 3: SafetyNet API Calls
Oct 04, 2013	Joshua W. Burns	Added Appendix F, chapter 6 and 7
Oct 07, 2013	Laxminarayana Yepuri	Added content for inspection and added additional entities like zone etc.
Oct 07, 2013	Bryan Shepherd	Added Project, Zone, Company, Contact, CompanyLocation, ContactAddress, ApplicationUser entities.
Oct 08, 2013	Joshua W Burns	Separated Delete operation; updated Incident record and query, added ER diagram
Oct 09, 2013	Laxminarayana Yepuri	Modified content for Inspection record and added content for Observation record
Oct 09, 2013	Laxminarayana Yepuri	Added content for Inspection query
Oct 09, 2013	Joshua W. Burns	<>
Oct 09, 2013	Edward P. Leisman	Added FAQ entries
Oct 10, 2013	Laxminarayana Yepuri	Added Content for inspection type record
Oct 10, 2013	Joshua W Burns	Added GetID request, added notes about updating
Oct 10, 2013	Prasad Reddy	Added SOAP Implementation, Technical Requirement sections. Added zoneid to login request and version to login response.
Oct 10, 2013	Bryan Shepherd	Adding additional Project Entities and table cleanup.
Oct 11, 2013	Prasad Reddy	Added Security details, version to End Point urls
Oct 11, 2013	Bryan Shepherd	Updated some Project information.
Oct 14, 2013	Joshua W Burns	Updated a few sections per Robb' comments, added a Deleting Data section to Chapter 2.
Oct 16, 2013	Joshua W Burns	Updated Chapter 3; added FAQ for Which Request to Use; added Chapter 2 – Updating Data section
Oct 17, 2013	Joshua W Burns	Updated responses for insert and update
Oct 23, 2013	Joshua W Burns	Updated FindByID request and response to only include one id/return one record
Oct 29, 2013	Joshua W Burns	Updated roadmap table; removed Login and Logout operations
Nov 4, 2013	Joshua W Burns	Refactored chapter 5 record sections, added Common Record Fields section; added Appendix H: Zone Specific Configuration
Nov 5, 2013	Teresa Lutz	Updated necessary fields by record
Nov 6, 2013	Joshua W Burns	Updated IncidentQuery to use date, updated format of inspection record table; added date and datetime data types
Nov 7, 2013	Joshua W Burns	SNQA-198 – removed resultCnt from FindByID and updated diagram
Nov 8, 2013	Joshua W Burns	SNQA-215 - update references to modifyDate
Nov 11, 2013	Joshua W Burns	SNQA-242 – update datetime format to match validation in XSD
Nov 14, 2013	Joshua W Burns	Add Observation FindByID footnote to chapter 1
Nov 15. 2013	Joshua W Burns	SNQA-259 – hidePercentSafeFlag is required for

		contact
Nov 19, 2013	Joshua W Burns	SNQA-303 – correct maxResults to 3000
Nov 21, 2013	Joshua W Burns	Update inspection record to match XSD
Nov 26, 2013	Joshua W Burns	SNQA-358 – change projectGroupCode and teamRoleCode in ProjectTeamAssociation record; SNQA-344 – remove commentImage from Inspection and Observation; SNQA-344 – clean up observation.
Dec 06, 2013	Prasad Reddy	Added section “Enable Web Services”
Jan 22, 2014	Edward P Leisman	Included Griffin’s modifications
Sep 25, 2014	Joshua W Burns	Update query request
Sep 25, 2014	Joshua W Burns	Update Inspection Query with new fields from SNWS-88.
Nov 19, 2014	Joshua W Burns	Initial update for WS 1.1
August 25, 2015	Joshua W Burns	Add updates for 1.1.7 – new ConfigServiceEndpoint and operation
May 5, 2017	Joshua W Burns	Removed specific record chapters, WSDL can be used to get the same information
June 21, 2017	Joshua W Burns	Added image/attachment section, paging, restrictions
November 11, 2017	Joshua W Burns	Updates per discussion with customer

Chapter 1: Overview

This document can be used to implement data integration with Predictive Solutions Corporation's SafetyNet Web Services application.

Intended Audience

This document is intended for Information Technology (IT) professionals and software developers who develop software applications to push data to, and/or pull data from the SafetyNet Web Service. As such, it is technical in nature, and assumes that the reader has some knowledge of the SafetyNet application and business domain and technical understanding of web services.

About this Document

This document is divided into seven chapters:

Throughout this document, there may be information that is important to the user of the SafetyNet Web Service. This can include technical limitations, warnings or other relevant information. This information is called out using the following convention:

This is important information about the SafetyNet Web Service. Please read!

References

The following are references to related documents or specifications used in this document.

SOAP 1.1 Description	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
Web Services Security UsernameToken Profile 1.0	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
XSD Date and Time Data Types	https://www.w3schools.com/xml/schema_dtypes_date.asp

Acronyms and Abbreviations

XML	Extensible Markup Language
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
XSD	XML Schema Definition

Scope of the Web Service

- Batch requests are not supported. Each request can only contain one request type. For instance, sending a request with an update, an insert and a delete is not supported. Three separate requests must be made by the client to satisfy this requirement.
- Only one record can be inserted/updated/deleted per request. It is the responsibility of the client to send multiple requests when inserting/updating more than one record. If multiple

records are included in these requests, an error code will be returned. There are two exceptions:
    - multiple observations can be inserted with an inspection or with the standalone observation insert operation
    - multiple record associations can be inserted, updated or deleted in a single request
- Only one id can be sent in a FindByID request. It is the responsibility of the client to send multiple requests to retrieve records using multiple ids. If multiple ids are included in these requests, an error code will be returned.
- The maximum number of results for a Query Request is 3000. The GetIDs request is not limited as it will return all ids.
- Every request must contain authentication information. See *Error! Reference source not found.* for more information on authentication.
- Maximum file size of an attachment is 2MB.
- When querying for Inspections, only summary data for observations will be returned. To get the observation data for an inspection, use the inspection FindByID operation.
- The maximum number of observations that are returned with an inspection using the inspection FindByID operation is 10,000.
- The maximum number of observations that can be inserted at one time (with inspection insert or observation insert) is 500.

Supported Record Types and Operations

The SafetyNet Web Service supports the following record types:

- Action Items
- Company
- Contact
- Incident
- Inspection
- Observation
- Project
- User

The following operations are supported for the above record types:[1]

- Query: Return full records that match criteria
- FindByID: Return full record that matches an id
- GetIDs: Returns only the IDs of records that match criteria
- Count: Get the total number of records s that match criteria
- Insert: Adds a specified record
- Update: Updates a specified record
- Delete: Deletes a specified record

[1] Inspection service contains these operations for both inspection and observation entities.

- Codes: Returns the zone specific lookups used for records (needed when inserting and updating an records)

The following operations are available for Project and Company records:

- Association Query: Return associations for a particular record
- Association Insert: Adds associations for a particular record
- Association Update: Updates associations
- Association Delete: Deletes associations

Additional operations are available for specific services. See Chapter 4: Specific Operations and Functionality and

Appendix D: Web Service Versions for more information.

Chapter 2: Getting Started

Enabling Web Services

SafetyNet must be configured to enable web services for a client. This can be done by the process improvement manager for the specific client. If a client tries to use web services and it has not been enabled, user authentication will fail.

SOAP Implementation

The SafetyNet web service implements SOAP version 1.1 for request and responses. See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ for a detailed description of SOAP version 1.1.

Endpoint URLs

All SafetyNet API calls must be contained in an HTTP POST request with the WS Security SOAP header as outlinedin the ***Authentication*** section below. Each SOAP request will be made a URL in the format:

https://ws.predictivesolutions.com/service/soap/<name>Service

where <name> is one of the record types listed in Chapter 1.

The WSDL page also contains the endpoint URLs for each service. Note that the endpoint URL is the same regardless of what version of the service is used.

WSDL

Web Services Description Language (WSDL) is a standardized format used to describe the web service. WSDLs are available by accessing the following URL:

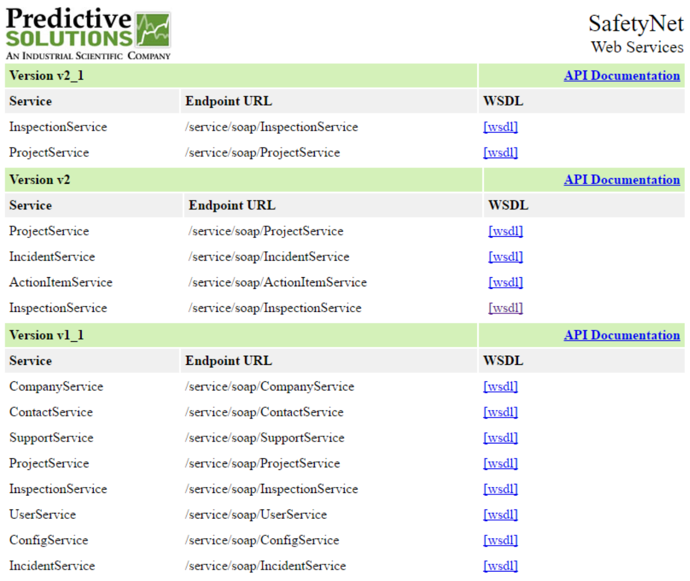
https://ws.predictivesolutions.com/service/soap/<name>Service_<version>.wsdl

where <name> is one of the record types listed in Chapter 1

and <version> is one of the supported versions of the record.

Using the WSDL, developers can generate proxy classes in a programming language such as Java or C#. This document describes the web service, WSDL and usage of that service. Details on generating client code and proxy classes are not provided in this document.

A WSDL page is provided that lists the available WSDLs for the web service and can be accessed via the URL https://ws.predictivesolutions.com/service/.



Predictive SOLUTIONS
An Industrial Scientific Company

SafetyNet
Web Services

Version v2_1		API Documentation
Service	Endpoint URL	WSDL
InspectionService	/service/soap/InspectionService	[wsdl]
ProjectService	/service/soap/ProjectService	[wsdl]

Version v2		API Documentation
Service	Endpoint URL	WSDL
ProjectService	/service/soap/ProjectService	[wsdl]
IncidentService	/service/soap/IncidentService	[wsdl]
ActionItemService	/service/soap/ActionItemService	[wsdl]
InspectionService	/service/soap/InspectionService	[wsdl]

Version v1_1		API Documentation
Service	Endpoint URL	WSDL
CompanyService	/service/soap/CompanyService	[wsdl]
ContactService	/service/soap/ContactService	[wsdl]
SupportService	/service/soap/SupportService	[wsdl]
ProjectService	/service/soap/ProjectService	[wsdl]
InspectionService	/service/soap/InspectionService	[wsdl]
UserService	/service/soap/UserService	[wsdl]
ConfigService	/service/soap/ConfigService	[wsdl]
IncidentService	/service/soap/IncidentService	[wsdl]

Figure 1: The SafetyNet Web Service WSDL page

There are multiple versions available for the web service, and not all versions provide a service for each of the core records. This allows flexibility for clients in that they can choose to upgrade to a more recent version of a specific service to take advantage a new features, or remain at an existing version until they decide to upgrade. Note that different versions of the web service are not compatible with each other, the schema and validation will be different and therefore clients will need to update the client code as well.

This document does not detail each service in terms of specific fields, types and validation rules. The WSDL provides this information and can be used for that purpose.

Security

SafetyNet web services implements the OASIS WS Security specification in order to authenticate and identify users for a request. This means that every SOAP request must contain a WS Security element in the header of the message, utilizing the UserNameToken profile as described in the specification.

Since SafetyNet web services uses SSL, this information will be encrypted when travelling over the Internet.

For more details on the OASIS WS Security Specification, see https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Authentication

The following is an example of the SOAP header that is used for authentication:

```
<soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
            xmlns:wsse="../oasis-200401-wss-wssecurity-secext-1.0.xsd"
            xmlns:wsu="../oasis-200401-wss-wssecurity-utility-1.0.xsd">
```

```
        <wsse:UsernameToken wsu:Id="UsernameToken-7">
            <wsse:Username>username</wsse:Username>
            <wsse:Password
                        Type="../oasis-200401-wss-username-token-profile-
1.0#PasswordText">password</wsse:Password>
        </wsse:UsernameToken>
    </wsse:Security>
</soapenv:Header>
```

Note: namespaces and schema type attributes in the above example have been abbreviated for clarity. The abbreviated path is: http://docs.oasis-open.org/wss/2004/01.

In the example above, the username token must contain a username and password element.

- username – this identifies the user in SafetyNet. It can be one of two formats:
    - username@zoneID, where zoneID is a 5 digit zone number for the user
      ex: admin@10003
    - username@zoneURL, where zoneURL is the fully qualified URL of the user's company[2] in SafetyNet
      ex: admin@dpr.predictivesolutions.com
- password – the password of the user (remember request will be encrypted)

The underlying authentication service will authenticate the user and return an error if any of the following is true:

- username is incorrect
- password is incorrect
- web service is not enabled for user's zone
- user has not agreed to EULA[3]
- user is locked
- contact for user is inactive
- zone is inactive

For security reasons, a general error message will be returned for any of the above cases:

```
<soapenv:Envelope>
   <soapenv:Header></soapenv:Header>
   <soapenv:Body>
      <ContactCodesResponse>
         <responseCode>1</responseCode>
         <message>Invalid Username Password Pair</message>
         <details>...</details>
      </ContactCodesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

[2] company refers to the company named created for a zone url mapping in the SafetyNet system. If you are unsure of what the value should be, contact support. This value is also the subdomain used when accessing the SafetyNet web application.
[3] This feature is configurable on a per zone basis and may not apply to all zones

Authorization

SafetyNet web services uses the same role based authorization that of web application. If the user does not possess the appropriate privileges for a request, the web service will return an error.

Testing

For testing purposes, please use the below url which points to our `sandbox` environment. PLEASE develop and test against this environment prior to Production release:

https://sandbox.predictivesolutions.com/service/soap/<name>Service

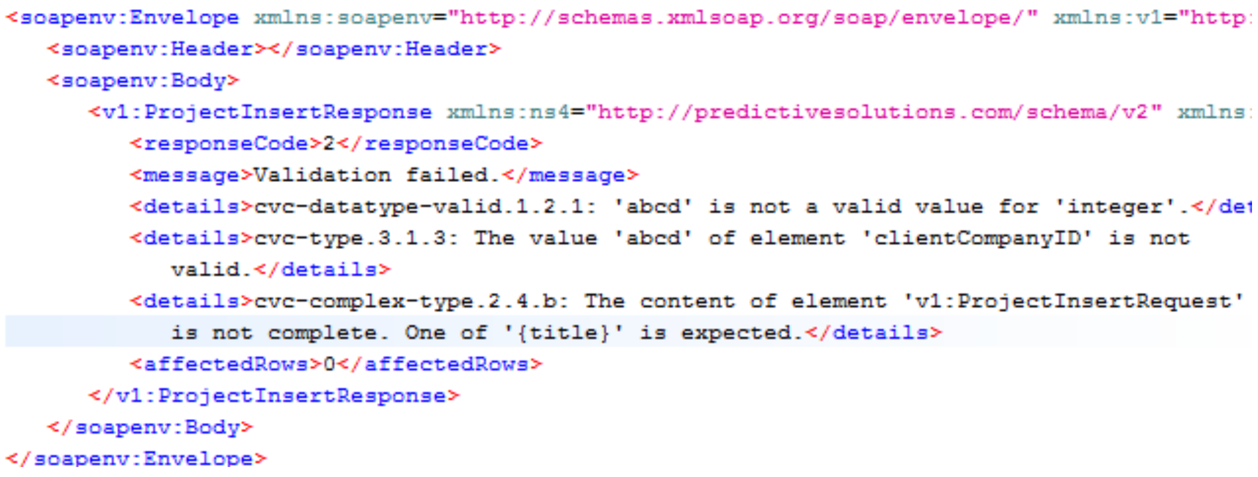
where <name> is one of the records listed in Chapter 1.

Validation

Validation of requests sent to the SafetyNet Web Service is done in two parts.

Schema Validation

First, the XML message itself is validated against the schema. If the request is missing a field or has an invalid value for a field, a validation message will be returned. In the example below, the ProjectInsertRequest omitted the <title> element and provided an invalid value for <clientCompanyID>[4]:

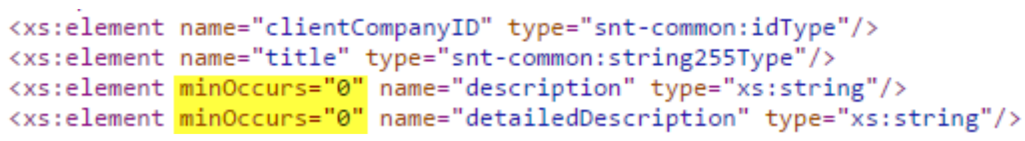


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http:
   <soapenv:Header></soapenv:Header>
   <soapenv:Body>
      <v1:ProjectInsertResponse xmlns:ns4="http://predictivesolutions.com/schema/v2" xmlns:
         <responseCode>2</responseCode>
         <message>Validation failed.</message>
         <details>cvc-datatype-valid.1.2.1: 'abcd' is not a valid value for 'integer'.</de
         <details>cvc-type.3.1.3: The value 'abcd' of element 'clientCompanyID' is not
            valid.</details>
         <details>cvc-complex-type.2.4.b: The content of element 'v1:ProjectInsertRequest'
            is not complete. One of '{title}' is expected.</details>
         <affectedRows>0</affectedRows>
      </v1:ProjectInsertResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 2: Example of failed schema validation

The WSDL designates what fields are optional using the minOccurs indicator. For the request in the example above, note how the <title> and <clientCompanyID> do not have minOccurs, while other fields have "minOccurs=0". The default if not specified is "minOccurs=1", which designates that the field is required.

[4] Generated proxy classes would not allow the assignment of a string to an int; this example uses direct SOAP messages and is provided for the developer's benefit. In most cases, the generated classes will not allow wrong types to be used.



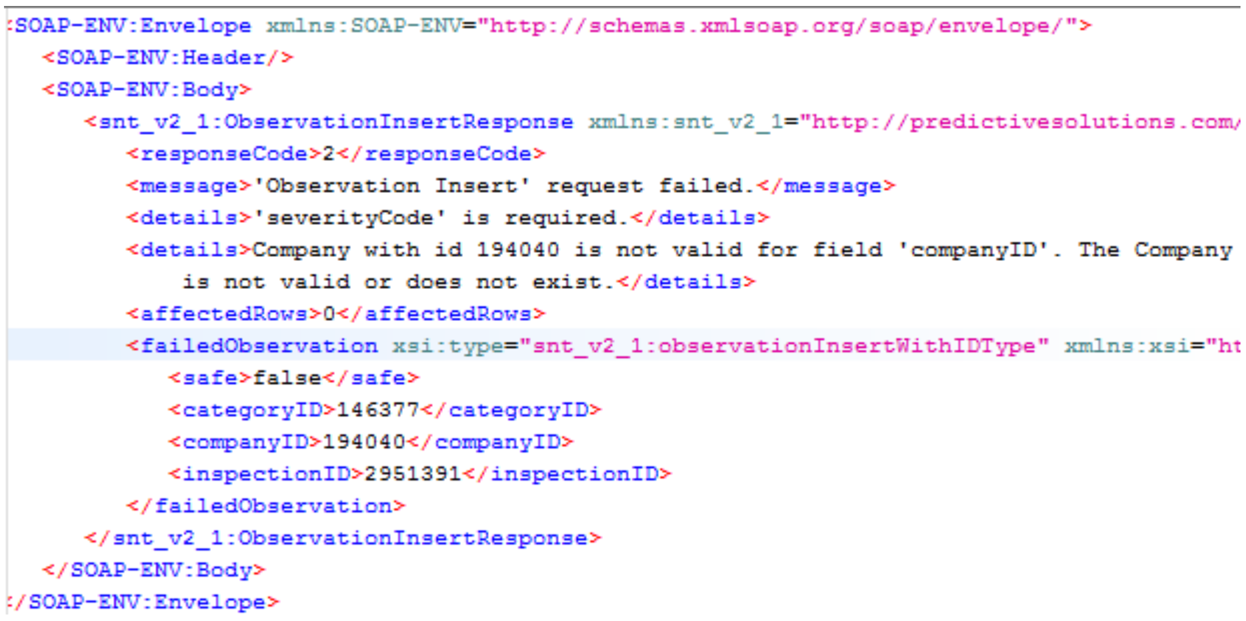
```
<xs:element name="clientCompanyID" type="snt-common:idType"/>
<xs:element name="title" type="snt-common:string255Type"/>
<xs:element minOccurs="0" name="description" type="xs:string"/>
<xs:element minOccurs="0" name="detailedDescription" type="xs:string"/>
```

:

Fields specified as required in schema apply to ALL zones in SafetyNet. Specific required fields are validated in the second step as detailed in the next section.

SafetyNet Validation

Once a message has passed schema validation, the request is validated against SafetyNet specific rules, such as required fields that are specific to each customer.



```
:SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v2_1:ObservationInsertResponse xmlns:snt_v2_1="http://predictivesolutions.com/
         <responseCode>2</responseCode>
         <message>'Observation Insert' request failed.</message>
         <details>'severityCode' is required.</details>
         <details>Company with id 194040 is not valid for field 'companyID'. The Company
             is not valid or does not exist.</details>
         <affectedRows>0</affectedRows>
         <failedObservation xsi:type="snt_v2_1:observationInsertWithIDType" xmlns:xsi="ht
            <safe>false</safe>
            <categoryID>146377</categoryID>
            <companyID>194040</companyID>
            <inspectionID>2951391</inspectionID>
         </failedObservation>
      </snt_v2_1:ObservationInsertResponse>
   </SOAP-ENV:Body>
:/SOAP-ENV:Envelope>
```

Figure 3: Example of failed validation rules

In the example above, <severityCode> is required. However, it's not specified as required in the schema because its only required for unsafe observations, and the request in this example is sending an unsafe. The response also returns a validation message about the specified company "is not valid or does not exist"; before inserting the observation the ids of any associated records (such as company in this case) are also validated to ensure integrity of the data.

The following is a list of validation checks that SafetyNet performs before executing the requested operation:

- ids of associated records are valid (they exist, are active, and belong to the zone of the user sending the request)
- required fields are set (see Required Fields)
- customer specific rules are enforced (require comments for unsafe observations, use project segment, etc.)

lookup codes are correct (see Response

The Delete response contains the result of the delete request.   In addition to the fields outlined ***Response***

The Codes response contains all of the lookup codes for the specific record.   These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Delete response contains the following fields:

Field	Type	Description
affectedRows	int	The number of rows that were affected by the insert operation. This will always be 0 (zero) if the insert failed, or 1 (one) if the insert was a success.

Figure 21: Delete Response Fields

- Codes)
- any other checks for conditions that may cause data integrity issues, conflicts between fields, etc.

Deleting Data

SafetyNet employs a “soft delete”, that is the underlying data is not deleted from the database but rather a column is updated to designate whether the record is active or not. Clients can delete data from the SafetyNet database by using the Delete Operation.

When using methods that query the database (Query, FindByID, Count, GetIDs), records marked as deleted will be ignored. In other words, only records that are active (not deleted) are returned in a response. When using the FindByID method, if an id of a deleted record is specified, an error code will be returned.

There is no operation to undo a delete operation, so exercise caution when using this method. To undo a delete to a record, you must contact SafetyNet Support.

To prevent deleting of records using old versions of the data, each delete request must specify the modifyDate field, and the value of this field must match the database record before the delete occurs. If they do not match, an error is returned and the record is not updated. To get the correct modifyDate for an record, a user should fetch the record to be updated using the FindByID operation. This ensures that the user has retrieved the most recent version of the record before updating it.

A delete request will result in an error and the record will not be updated if:

- the id of the record is missing or not valid
- the modifyDate of the record in the delete request is missing
- the modifyDate of the record in the delete request does not match the record in the database

Updating Data

Records can be updated using the update request of the web service. This request takes a specific record type with the id field set. SafetyNet assumes that the client is the authoritative source of data; therefore the web service does not make any assumptions or decisions when an update request is made.

When performing an update, the full record must be sent. The web service does not support partial updates. Whatever data is sent in the request will be updated in the database unconditionally (with the exception of updates that would cause an error). For example, if project title is not sent in an update request, it will be cleared in the databse.

To prevent updating of records using old versions of the data, each update request must specify the modifyDate field, and the value of this field must match the database record before the update occurs. If they do not match, an error is returned and the record is not updated. To get the correct modifyDate for a record, a user should fetch the record to be updated using the FindByID request. This ensures that the user has retrieved the most recent version of the record before updating it.

Missing fields or fields with an empty value are treated the same by the web service, the field will be cleared in the database. **Sending update requests with missing fields can lead to loss of data.**

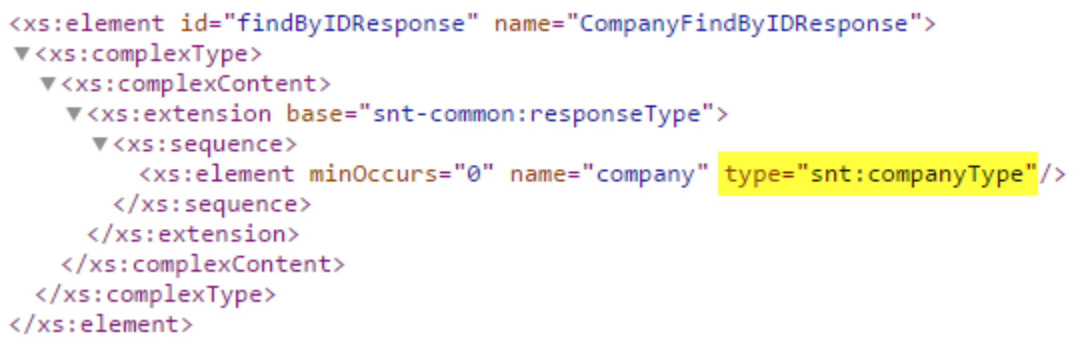
An update request will result in an error and the record will not be updated if:

- the id of the record is missing or not valid
- any required fields for the record are missing or contain empty values
- the modifyDate of the record in the update request is missing
- the modifyDate of the record in the update request does not match the record in the database

A Note About Record Type Restrictions

The SafetyNet Web Service WSDLs use the idea of a restriction to define a reusable record type and define different usage rules for specific operations. This is important to note because when proxy classes are generated using the WSDL, these restriction rules are not preserved in the generated classes.

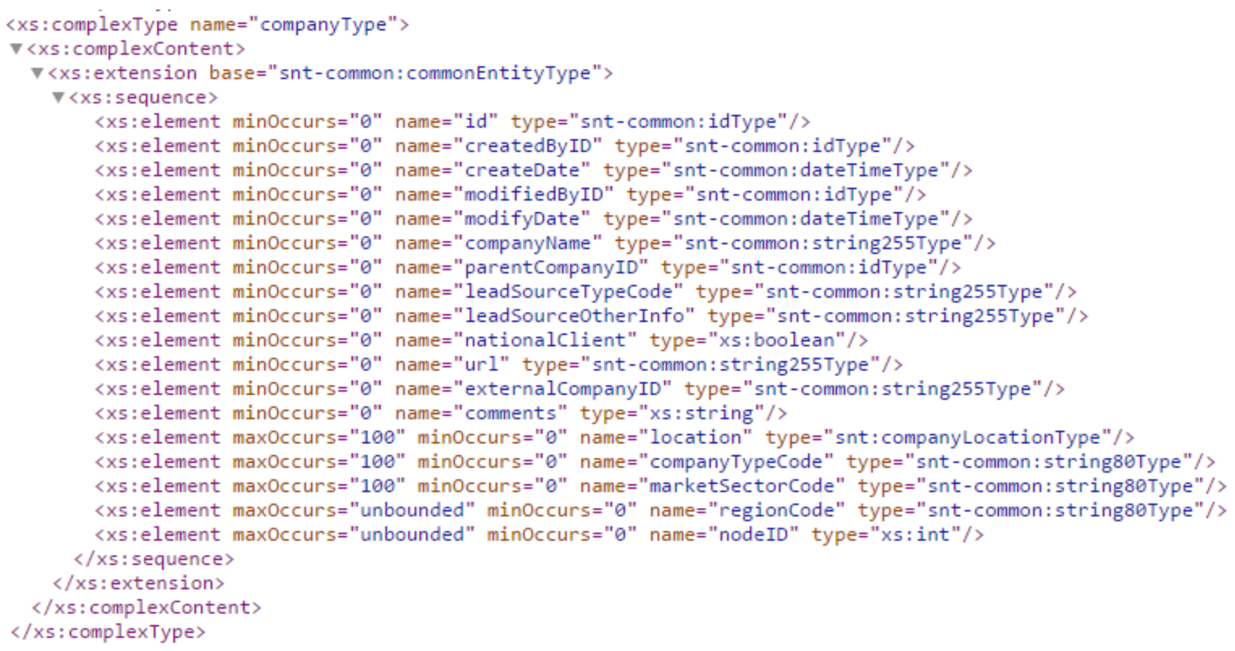
To illustrate this point, the company FindByID operation returns a company, which uses snt:companyType as its type.



```
<xs:element id="findByIDResponse" name="CompanyFindByIDResponse">
 ▼<xs:complexType>
   ▼<xs:complexContent>
     ▼<xs:extension base="snt-common:responseType">
       ▼<xs:sequence>
           <xs:element minOccurs="0" name="company" type="snt:companyType"/>
         </xs:sequence>
       </xs:extension>
     </xs:complexContent>
   </xs:complexType>
 </xs:element>
```

Figure 4: WSDL definition for CompanyFindByIDResponse

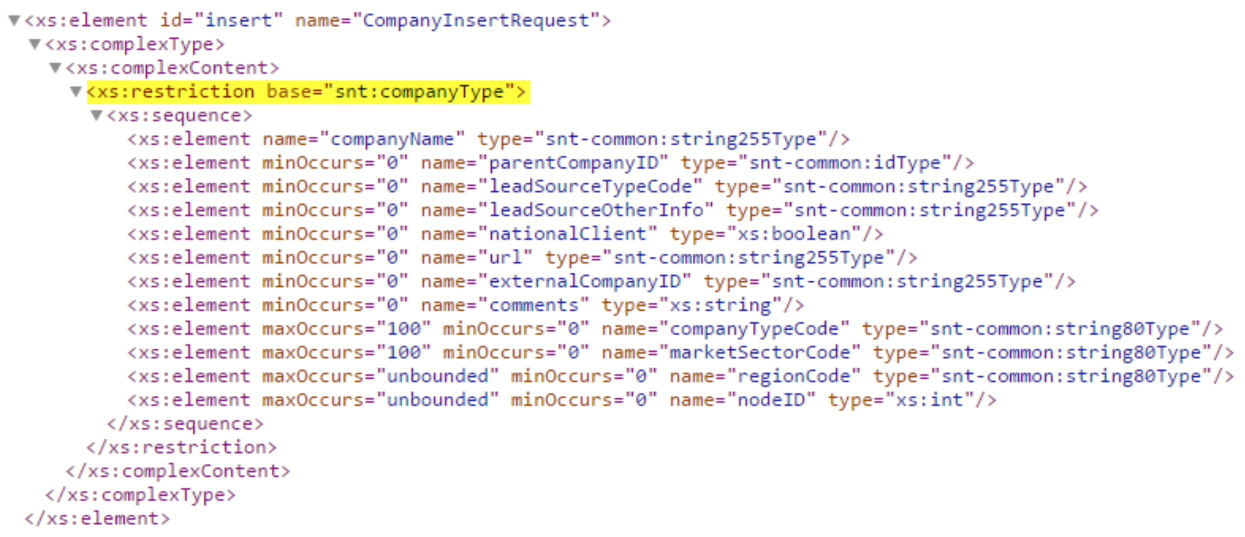
This type defines a company record in SafetyNet; it contains fields that describe the specific record in the database and is used by multiple operations (Query, FindByID, Insert, and Update).



```
<xs:complexType name="companyType">
▼<xs:complexContent>
  ▼<xs:extension base="snt-common:commonEntityType">
    ▼<xs:sequence>
        <xs:element minOccurs="0" name="id" type="snt-common:idType"/>
        <xs:element minOccurs="0" name="createdByID" type="snt-common:idType"/>
        <xs:element minOccurs="0" name="createDate" type="snt-common:dateTimeType"/>
        <xs:element minOccurs="0" name="modifiedByID" type="snt-common:idType"/>
        <xs:element minOccurs="0" name="modifyDate" type="snt-common:dateTimeType"/>
        <xs:element minOccurs="0" name="companyName" type="snt-common:string255Type"/>
        <xs:element minOccurs="0" name="parentCompanyID" type="snt-common:idType"/>
        <xs:element minOccurs="0" name="leadSourceTypeCode" type="snt-common:string255Type"/>
        <xs:element minOccurs="0" name="leadSourceOtherInfo" type="snt-common:string255Type"/>
        <xs:element minOccurs="0" name="nationalClient" type="xs:boolean"/>
        <xs:element minOccurs="0" name="url" type="snt-common:string255Type"/>
        <xs:element minOccurs="0" name="externalCompanyID" type="snt-common:string255Type"/>
        <xs:element minOccurs="0" name="comments" type="xs:string"/>
        <xs:element maxOccurs="100" minOccurs="0" name="location" type="snt:companyLocationType"/>
        <xs:element maxOccurs="100" minOccurs="0" name="companyTypeCode" type="snt-common:string80Type"/>
        <xs:element maxOccurs="100" minOccurs="0" name="marketSectorCode" type="snt-common:string80Type"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="regionCode" type="snt-common:string80Type"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="nodeID" type="xs:int"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 5: The CompanyType schema

The Insert operation also uses the same type however it specifies a “restriction”. This means that it only uses a subset of the fields from companyType.



```
▼<xs:element id="insert" name="CompanyInsertRequest">
  ▼<xs:complexType>
    ▼<xs:complexContent>
      ▼<xs:restriction base="snt:companyType">
        ▼<xs:sequence>
            <xs:element name="companyName" type="snt-common:string255Type"/>
            <xs:element minOccurs="0" name="parentCompanyID" type="snt-common:idType"/>
            <xs:element minOccurs="0" name="leadSourceTypeCode" type="snt-common:string255Type"/>
            <xs:element minOccurs="0" name="leadSourceOtherInfo" type="snt-common:string255Type"/>
            <xs:element minOccurs="0" name="nationalClient" type="xs:boolean"/>
            <xs:element minOccurs="0" name="url" type="snt-common:string255Type"/>
            <xs:element minOccurs="0" name="externalCompanyID" type="snt-common:string255Type"/>
            <xs:element minOccurs="0" name="comments" type="xs:string"/>
            <xs:element maxOccurs="100" minOccurs="0" name="companyTypeCode" type="snt-common:string80Type"/>
            <xs:element maxOccurs="100" minOccurs="0" name="marketSectorCode" type="snt-common:string80Type"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="regionCode" type="snt-common:string80Type"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="nodeID" type="xs:int"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

Figure 6: CompanyInsertRequest using a restriction

This is an important concept to understand, proxy classes will not preserve the restriction in these cases. As a developer, the generated code will allow fields like createdDate and id to be set on an insert request, as its using the generic companyType, and not the restriction. However, setting these fields will cause a validation error to occur.

Always check the WSDL to get the correct fields to use for a service. Proxy classes will not maintain this information.

Visual Studio Nuances

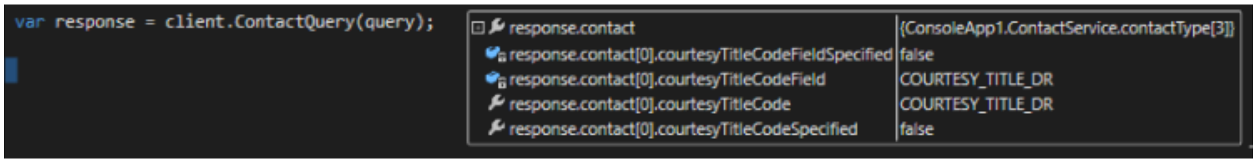
When using the SafetyNet API with Visual Studio, there are a few things to keep in mind when sending requests and processing responses.

Specified Fields

In order for Visual Studio to determine if a field is really set or has really been sent in a response, several fields are created in addition to the normal record fields. These Boolean fields add the suffix "isSpecified" to the field name. This is used to determine if a null value means that the value was explicitly set to null, or just was not set at all.

Specifically, fields that are enumeration fields in the response will default to the first value in the enumeration. However, the web service is not setting these values, the framework defaults them to a value as they cannot be null. To determine of these fields are actually set, you must check the "Specified" fields.

In the example below, "courtesyTitleCode" is set to "COURTESY_TITLE_DR", however the web service is sending null for this field. Visual Studio sets the courtesyTitleCodeSpecified to false in this case. Clients should guard against this case and not assume that the value is correct without looking at the additional fields.



```
var response = client.ContactQuery(query);
```

```
response.contact                                      {ConsoleApp1.ContactService.contactType[3]}
  response.contact[0].courtesyTitleCodeFieldSpecified false
  response.contact[0].courtesyTitleCodeField          COURTESY_TITLE_DR
  response.contact[0].courtesyTitleCode               COURTESY_TITLE_DR
  response.contact[0].courtesyTitleCodeSpecified      false
```

Overview

This chapter outlines the SafetyNet API calls, request and response formats. See '***Which Request Should I Use?'*** for more information on which request to use for particular usage scenarios.

Generic request and response formats are covered in this chapter. Specific record types (used for Update and Insert requests and returned in Query and FindByID responses) are documented in the specific WSDL for the service being used.

Common Response

All requests will generate a response which, regardless of the operation, will return common fields along with the actual data payload.

When contacting support, the response fields are extremely useful in troubleshooting issues. Therefore clients should include a mechanism to easily retrieve these fields in their application so they can be sent back to support if needed.

Field	Type	Description
responseCode	int	Status code returns a code that can be used to determine if the request was successful or not. A value of 0 (zero) designates that the request was successful. See **Response Codes** for more information.
message	string	A message indicating the specific reason a request failed, or a message indicating success
details	string	additional details about a failed message
additionalInfo	string	additional information that can be used for troubleshooting purposes.
response payload		The value of this field is dependent on the operation and record type. The response can be one of the following: • FindByID Reponse • Query Reponse • GetIDs Response • Count Response • Update/Insert Response • Delete Response • Codes Response • Association Query Response • Association Insert/Update Response • Association Delete Response

Figure 7: Response Fields

Response Codes

Every response from the web service will return a response code indicating if the request was successful or the error that was encountered if it was not. Some codes are specific to the operation that was requested. A status code of 0 indicates that the request was successful; any other value indicates that there was a problem with the request.

CODE	MESSAGE
0	Request completed successfully. This does not indicate that data was returned, only that the request returned from the server without an issue.
1	Request failed due to authentication. See ***Security*** for more information on causes of authentication errors.
2	Request failed due to validation issues. Details of the reason are included within the response, including the invalid fields and values. Validation errors are caused by one of four reasons: • Schema validation failed • Missing required fields • Using incorrect data • Failing validation logic See ***Validation*** for more information.
3	Request failed due to a processing error. Details of the reason are included within the response. If a user has insufficient privileges, this code is returned.

Query

Use the Query Request to find multiple records using the specified criteria.

Request

The fields common to all Query Requests are:

Field	Type	Description
resultsPerPage	int	This field specifies the maximum number of results that the query operation should return. This field can be used to improve performance of large queries. If not specified, the default is 500, and the maximum value is 3000. See ***Pagi*** for more information.
pageNumber	int	Specifies the page number to return in the result set for paginated result sets. Defaults to 1 (first page). See ***Pagi*** for more information.
criteria payload	record criteria	The criteria payload varies based on the requested record. Reference the WSDL for documentation on specific criteria fields.

Figure 8: Query Request Fields

Response

The Query response contains the records that match the criteria in the Query Request. In addition to the fields outlined in ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Query response contains the following fields:

Field	Type	Description
resultCount	int	The number of result records that are contained in the response. Note: this will be less than or equal to the maxResults field specified in the Query request.
totalCount	integer	The number of total results in the database. This field will be set only if the total is greater than resultCount
totalPages	integer	The total number of pages in the dataset. Basically, totalCount/resultsPerPage (from request) rounded to the next whole number.
record payload	record	For a Query Response, this will zero or more records. The specific record payload varies based on the requested record. Reference *the WSDL* for specific fields for each record.

Figure 9: Query Response Fields

FindByID

Use the FindByID Request when you already know the id of a single record you want to retrieve. The most common use case would be to call FindByID for a record before calling the Update Request.

Request

Unlike Query Requests, which have a specific criteria for each record type, the FindByID request only uses an id.

Due to performance impacts, in some cases the FindByID response will return additional fields for the record as compared to a query response. An example is images for observations, FindByID will return the images for the record while Query will not (as query can deal with a large set of data).

The FindByID request contains the following fields:

Field	Type	Description
id	int	An ID to match the specific record to. The Find request can only contain one id.

Figure 10: Find Request Fields

Response

The FindByID response contains the full record that matches the specified id in the FindByID Request. In addition to the fields outlined in ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the FindByID response contains the following fields:

Field	Type	Description
record payload	Record	For a FindByID Response, this will be a single record. The record payload varies based on the requested record. Reference *the WSDL* for specific fields for each record.

Figure 11: Find Response Fields

GetIDs

The GetIDs request retrieves the ids of all records that match the specified criteria.

Request

Field	Type	Description
criteria payload	record criteria	The criteria payload varies based on the requested record. Reference the WSDL for documentation on specific criteria fields.

Figure 12: Find Request Fields

Response

The GetIDs response contains the all of the ids of the records that match the criteria in the GetIDs Request. In addition to the fields outlined in ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the GetIDs response contains the following fields:

Field	Type	Description
resultCount	int	The number of result IDs that are contained in the response.
id	integer	A list of ids of the records that matched the criteria.

Figure 13: Query Response Fields

Note that the GetID method only returns ids, if the actual record data is desired use the Query Request or FindByID Request instead. GetID response is not limited to a maximum number of results as the query operation is; it will return all IDs that match the criteria.

Count

The Count Request retrieves the total number of records that match the specified criteria. It is advantageous to use the Count Request in situations where the client wants to quickly get the total number of records but doesn’t want to get the actual records themselves.

Request

The Count request contains the following fields:

Field	Type	Description
query payload	record criteria	The criteria payload varies based on the requested record. Reference the WSDL for documentation on specific criteria fields.

Figure 14: Count Request Fields

Response

The Count response only contains the count of records that match the specified criteria. It can be used to quickly get a total count of a specific record, as it does not return any data. In addition to the fields outlined in ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Count response contains the following fields:

Field	Type	Description
resultCount	int	The number of records that matched the criteria.

Figure 15: Count Response Fields

Note that this response is similar to the Query and Find responses except that it does not actually contain results (a list of records).

Insert

The Insert request allows a user to create a new record.

Request

The Insert request contains the following fields:

Field	Type	Description
record payload	Record	The record payload varies based on the requested record. Reference *the WSDL* for specific fields for each record.

Figure 16: Update/Insert Request Fields

Note: The ID field of the record must **not** be set on an insert request.

Only one record can be inserted in a request. If multiple records are included in the request, an error will be returned.

Associations for a record, like a company location or project team must be inserted/updated in a separate request. See the Association operations section later in this chapter for more information.

Response

The Insert response contains the result of an insert operation. In addition to the fields outlined ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Update/Insert response contains the following fields:

Field	Type	Description
affectedRows	int	The number of rows that were affected by the insert operation. This will always be 0 (zero) if the insert failed, or 1 (one) if the insert was a success.
id	integer	The id of the inserted record

Figure 17: Insert Response Fields

Update

The Update request allows a user to update an existing record.

Request

SafetyNet assumes that the client is the authoritative source of data; therefore the web service does not make any assumptions or decisions when an update request is made. Whatever data is sent in the request will be updated in the database unconditionally. See *Updating Data* for important information regarding updating of records.

The Update/Insert request contains the following fields:

Field	Type	Description
record payload	Record	The record payload varies based on the requested record. Reference *the WSDL* for specific fields for each record.

Figure 18: Update/Insert Request Fields

Note: The ID field of the record **must be set** on an update request.

Only one record can updated in a request. If multiple records are included in the request, an error code will be returned.

Associations for a record, like a company location or project team must be updated in a separate request. See the Association operations section later in this chapter for more information.

Response

The Update response contains the result of an update operation. In addition to the fields outlined ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Update response contains the following fields:

Field	Type	Description
affectedRows	int	The number of rows that were affected by the insert operation. This will always be 0 (zero) if the insert failed, or 1 (one) if the insert was a success.

Figure 19: Update Response Fields

Delete

The delete operation deletes a record from the system. This is a soft delete, the record is marked as deleted but not removed from the system.

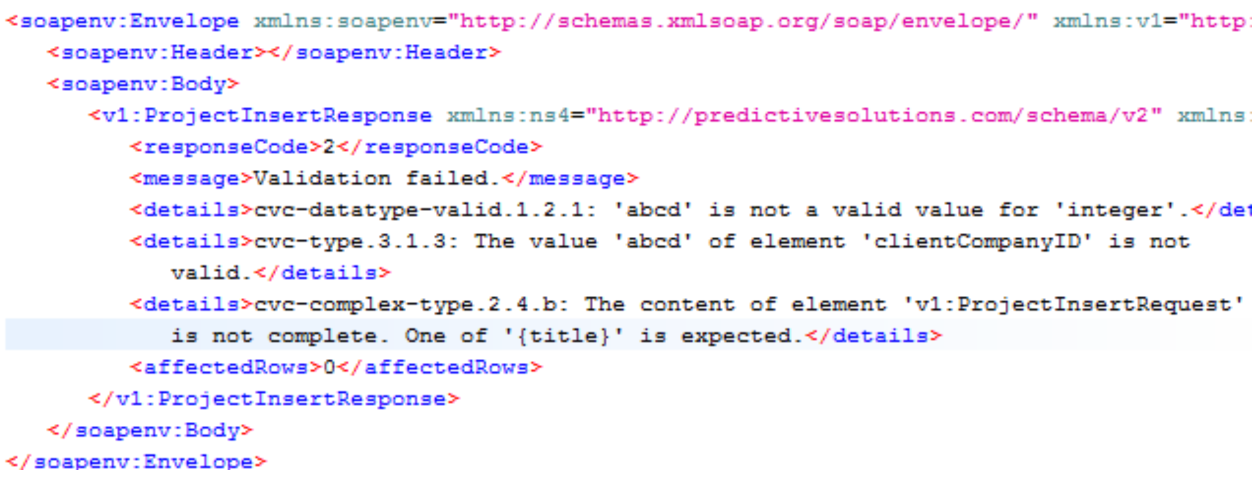
Request

The web service does not support an undelete operation, so use this request with See Validation

Validation of requests sent to the SafetyNet Web Service is done in two parts.

Schema Validation

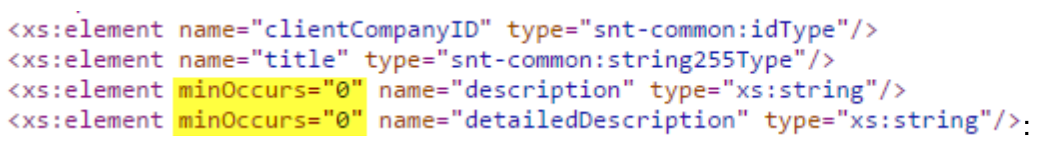
First, the XML message itself is validated against the schema. If the request is missing a field or has an invalid value for a field, a validation message will be returned. In the example below, the ProjectInsertRequest omitted the <title> element and provided an invalid value for <clientCompanyID>:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http:
   <soapenv:Header></soapenv:Header>
   <soapenv:Body>
      <v1:ProjectInsertResponse xmlns:ns4="http://predictivesolutions.com/schema/v2" xmlns:
         <responseCode>2</responseCode>
         <message>Validation failed.</message>
         <details>cvc-datatype-valid.1.2.1: 'abcd' is not a valid value for 'integer'.</de
         <details>cvc-type.3.1.3: The value 'abcd' of element 'clientCompanyID' is not
            valid.</details>
         <details>cvc-complex-type.2.4.b: The content of element 'v1:ProjectInsertRequest'
            is not complete. One of '{title}' is expected.</details>
         <affectedRows>0</affectedRows>
      </v1:ProjectInsertResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 2: Example of failed schema validation

The WSDL designates what fields are optional using the minOccurs indicator. For the request in the example above, note how the <title> and <clientCompanyID> do not have minOccurs, while other fields have “minOccurs=0”. The default if not specified is “minOccurs=1”, which designates that the field is required.

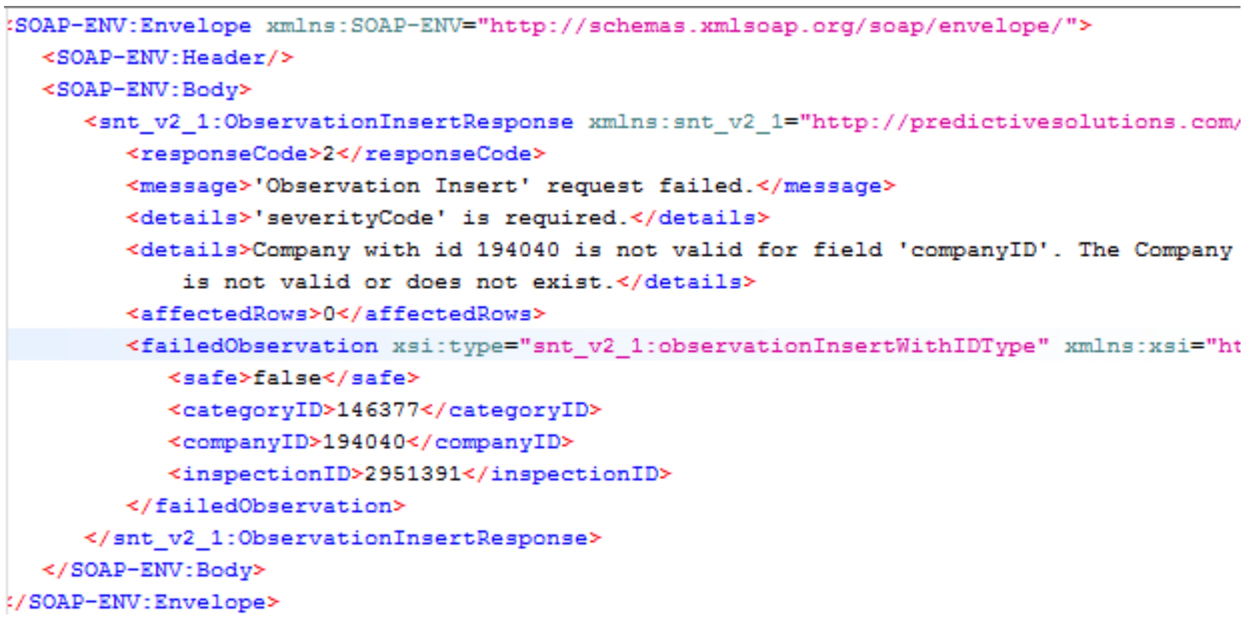


```
<xs:element name="clientCompanyID" type="snt-common:idType"/>
<xs:element name="title" type="snt-common:string255Type"/>
<xs:element minOccurs="0" name="description" type="xs:string"/>
<xs:element minOccurs="0" name="detailedDescription" type="xs:string"/>:
```

Fields specified as required in schema apply to ALL zones in SafetyNet. Specific required fields are validated in the second step as detailed in the next section.

SafetyNet Validation

Once a message has passed schema validation, the request is validated against SafetyNet specific rules, such as required fields that are specific to each customer.



```
:SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v2_1:ObservationInsertResponse xmlns:snt_v2_1="http://predictivesolutions.com/
         <responseCode>2</responseCode>
         <message>'Observation Insert' request failed.</message>
         <details>'severityCode' is required.</details>
         <details>Company with id 194040 is not valid for field 'companyID'. The Company
             is not valid or does not exist.</details>
         <affectedRows>0</affectedRows>
         <failedObservation xsi:type="snt_v2_1:observationInsertWithIDType" xmlns:xsi="ht
            <safe>false</safe>
            <categoryID>146377</categoryID>
            <companyID>194040</companyID>
            <inspectionID>2951391</inspectionID>
         </failedObservation>
      </snt_v2_1:ObservationInsertResponse>
   </SOAP-ENV:Body>
:/SOAP-ENV:Envelope>
```

Figure 3: Example of failed validation rules

In the example above, <severityCode> is required. However, it's not specified as required in the schema because its only required for unsafe observations, and the request in this example is sending an unsafe. The response also returns a validation message about the specified company "is not valid or does not exist"; before inserting the observation the ids of any associated records (such as company in this case) are also validated to ensure integrity of the data.

The following is a list of validation checks that SafetyNet performs before executing the requested operation:

- ids of associated records are valid (they exist, are active, and belong to the zone of the user sending the request)
- required fields are set (see Required Fields)
- customer specific rules are enforced (require comments for unsafe observations, use project segment, etc.)

lookup codes are correct (see Response

The Delete response contains the result of the delete request. In addition to the fields outlined ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Delete response contains the following fields:

Field	Type	Description
affectedRows	int	The number of rows that were affected by the insert operation. This will always be 0 (zero) if the insert failed, or 1 (one) if the insert was a success.

Figure 21: Delete Response Fields

- Codes)
- any other checks for conditions that may cause data integrity issues, conflicts between fields, etc.

Deleting Data for important information before using the Delete Request.

The Delete request contains the following fields:

Field	Type	Description
id	int	An ID to match the specific record to. The Delete request can only contain one id.
modifyDate	dateTime	

Figure 20: Delete Request Fields

Only one record can be deleted in a request. If multiple records are included in the request, an error code will be returned.

Response

The Delete response contains the result of the delete request. In addition to the fields outlined ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Delete response contains the following fields:

Field	Type	Description
affectedRows	int	The number of rows that were affected by the insert operation. This will always be 0 (zero) if the insert failed, or 1 (one) if the insert was a success.

Figure 21: Delete Response Fields

Codes

Some records contain fields that use a Lookup Code, and these are required to create or update an record. The LookupCodes Request returns a list of these codes for a specific record.

Request

This request should be called before updating or inserting so that the correct values can be included in a record.

If a request to update or insert a record contains an invalid code, then an error code will be returned.

Field	Type	Description
filter	string	A specific code type to return in the response. A request can have multiple filters. This should match the name of the field in the record that the code is used for **Note that the filter value must be the exact name, wildcards and substrings are not supported.**

Response

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

Chapter 4: Specific Operations and Functionality

The WSDL should be the authoritative source of information on specific web service operations. This chapter describes some advanced operations and provides examples of these operations as well that may not be so straightforward in the WSDL.

Required Fields

There are two ways to determine the required fields for an insert or update. The first is in the WSDL itself, which applies to all customers. The second is to call the fields operation to get the required fields within the context of a specific customer.

The WSDL will show fields that are required across all zones, but not specific to a zone. SO in order to get all fields that re required, this request must be used. Sending a request without a required field will generate a validation error.

The Fields response contains all of the required fields for each record type in SafetyNet. These fields are need to be provided for insert and update operations. In addition to the fields outlined in ***Response***

The Codes response contains all of the lookup codes for the specific record. These codes are used for insert and update operations. The LookupCodes Response contains the following fields:

Field	Type	Description
lookup payload	SimpleCodeGroupType	A collection of code groups for the specific record. See **Error! Reference source not found.** for more information on this type.

See ***Lookup Codes*** for more information.

InspectionCodes Response

For the inspection service, the codes response is different as it includes category and inspection type information.

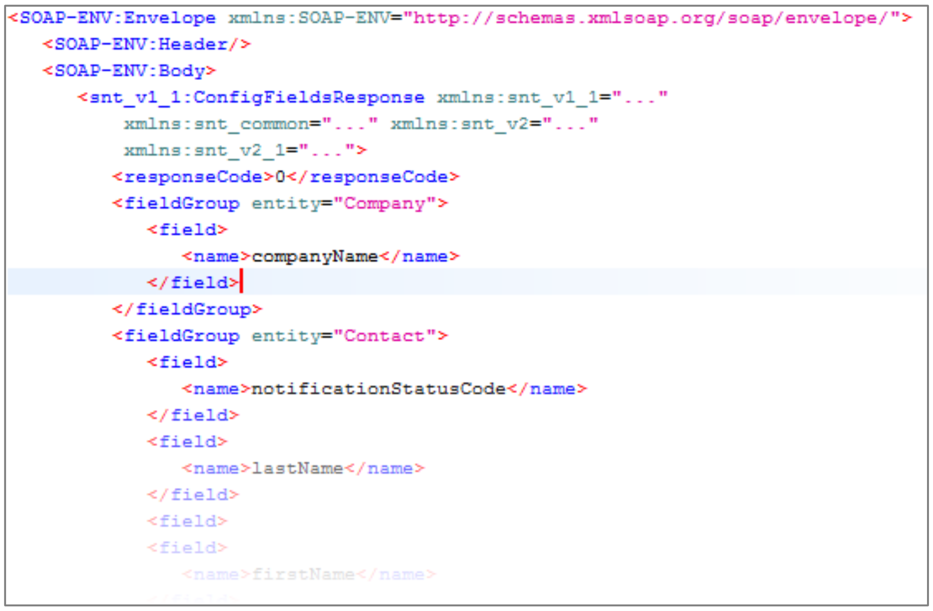
Field	Type	Description
lookup payload	inspectionCodesResponseType	The payload contains types, category and custom data information used when inserting or updating an inspection or observation.

Figure 22: Code response for Inspection Service

See ***Lookup Codes*** for more information.

, the Fields Response contains the following fields:

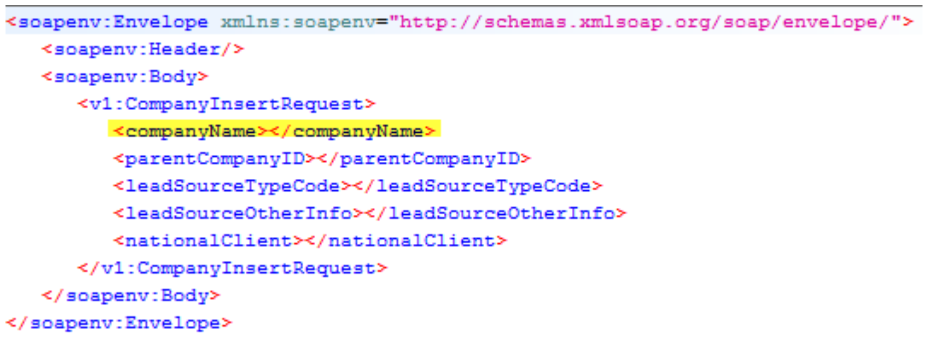
Field	Type	Description
fieldGroup payload	FieldGroupType	A collection of field groups for the specific record. See **Error! Reference source not found.** for more information on this type.



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v1_1:ConfigFieldsResponse xmlns:snt_v1_1="..."
          xmlns:snt_common="..." xmlns:snt_v2="..."
          xmlns:snt_v2_1="...">
         <responseCode>0</responseCode>
         <fieldGroup entity="Company">
            <field>
               <name>companyName</name>
            </field>
         </fieldGroup>
         <fieldGroup entity="Contact">
            <field>
               <name>notificationStatusCode</name>
            </field>
            <field>
               <name>lastName</name>
            </field>
            <field>
            <field>
               <name>firstName</name>
```

Figure 23: Required Fields Response

The name of a field returned in the fields response will match a field from that specific record.



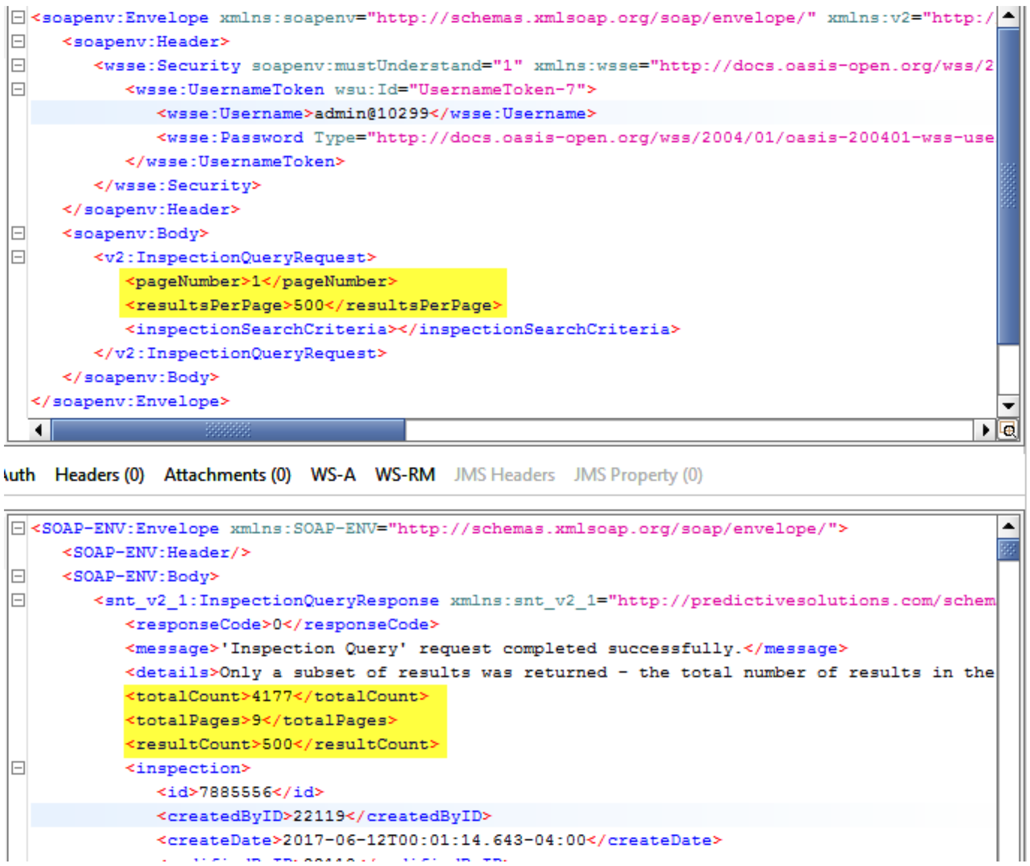
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <v1:CompanyInsertRequest>
         <companyName></companyName>
         <parentCompanyID></parentCompanyID>
         <leadSourceTypeCode></leadSourceTypeCode>
         <leadSourceOtherInfo></leadSourceOtherInfo>
         <nationalClient></nationalClient>
      </v1:CompanyInsertRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 24: Company insert request with required field

Pagination

For large data sets, the query requests will only return a subset of the records, known as a page. Additional requests must be made to the web service to fetch additional pages of records.

Each query request contains a <pageNumber> and <resultsPerPage> field. This allows the request to control how many records are sent back in each response. In the example below, the request is asking for the first 500 records of data that match the criteria.



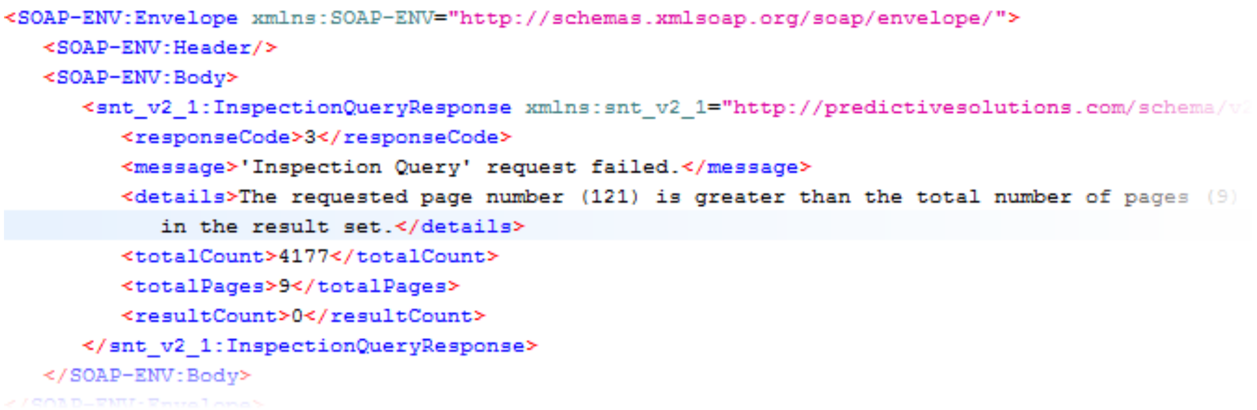
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v2="http:/
   <soapenv:Header>
      <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2
         <wsse:UsernameToken wsu:Id="UsernameToken-7">
            <wsse:Username>admin@10299</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-use
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body>
      <v2:InspectionQueryRequest>
         <pageNumber>1</pageNumber>
         <resultsPerPage>500</resultsPerPage>
         <inspectionSearchCriteria></inspectionSearchCriteria>
      </v2:InspectionQueryRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v2_1:InspectionQueryResponse xmlns:snt_v2_1="http://predictivesolutions.com/schem
         <responseCode>0</responseCode>
         <message>'Inspection Query' request completed successfully.</message>
         <details>Only a subset of results was returned - the total number of results in the
         <totalCount>4177</totalCount>
         <totalPages>9</totalPages>
         <resultCount>500</resultCount>
         <inspection>
            <id>7885556</id>
            <createdByID>22119</createdByID>
            <createDate>2017-06-12T00:01:14.643-04:00</createDate>
```

Figure 25: Query request

The response will contain no more than the specified <resultsPerPage> field (in this example 500) and will also contain the total number of pages that are required to represent the entire data set.

If the requested page number is greater than the total number of pages, an error is returned. The total number of records and pages are still returned in the response.



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v2_1:InspectionQueryResponse xmlns:snt_v2_1="http://predictivesolutions.com/schema/v2
         <responseCode>3</responseCode>
         <message>'Inspection Query' request failed.</message>
         <details>The requested page number (121) is greater than the total number of pages (9)
            in the result set.</details>
         <totalCount>4177</totalCount>
         <totalPages>9</totalPages>
         <resultCount>0</resultCount>
      </snt_v2_1:InspectionQueryResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

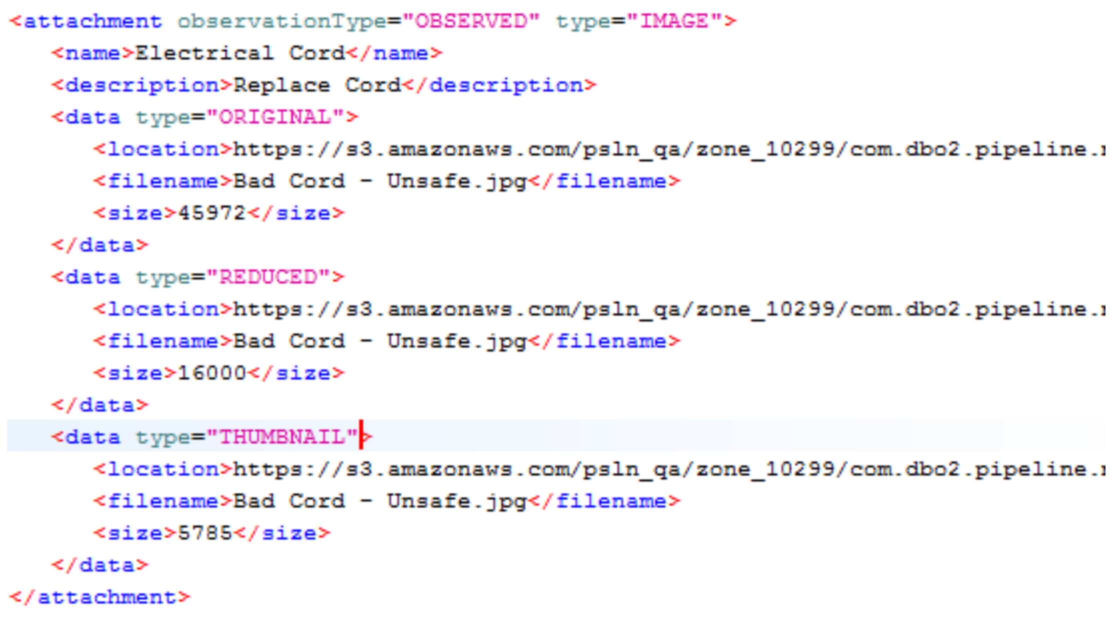
The maximum number of <resultsPerPage> for all query operations except observation query is 3000. Specifying a number larger than 3000 will result in a validation error. Observation query allows <resultsPerPage> to be set up to 10,000.

Images and Attachments

Inserting images and attachments is rather straightforward, however depending on the specific version of the service you are using the fields may be different.

InspectionService currently supports image images for observations. Images are specified using the <attachment> element in the request. The <attachment> element is also used in responses; for the request example below only the applicable fields are shown.

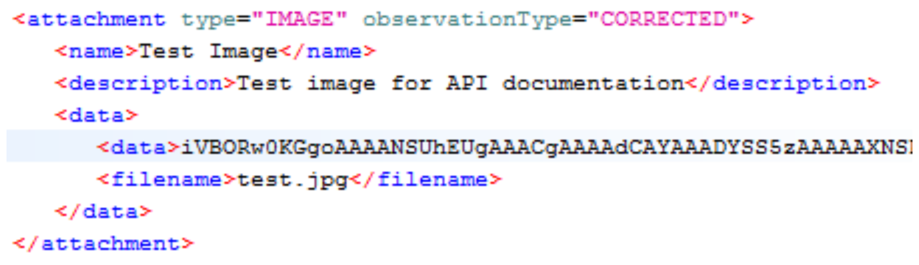
The attachment element contains additional fields in the response when using observation findByID:



```
<attachment observationType="OBSERVED" type="IMAGE">
   <name>Electrical Cord</name>
   <description>Replace Cord</description>
   <data type="ORIGINAL">
      <location>https://s3.amazonaws.com/psln_qa/zone_10299/com.dbo2.pipeline.
      <filename>Bad Cord - Unsafe.jpg</filename>
      <size>45972</size>
   </data>
   <data type="REDUCED">
      <location>https://s3.amazonaws.com/psln_qa/zone_10299/com.dbo2.pipeline.
      <filename>Bad Cord - Unsafe.jpg</filename>
      <size>16000</size>
   </data>
   <data type="THUMBNAIL">
      <location>https://s3.amazonaws.com/psln_qa/zone_10299/com.dbo2.pipeline.
      <filename>Bad Cord - Unsafe.jpg</filename>
      <size>5785</size>
   </data>
</attachment>
```

Figure 26: Attachment response for an image

To insert an image with an observation, send an <attachment> element:



```
<attachment type="IMAGE" observationType="CORRECTED">
   <name>Test Image</name>
   <description>Test image for API documentation</description>
   <data>
      <data>iVBORw0KGgoAAAANSUhEUgAAACgAAAAdCAYAAADYSS5zAAAAAXNS
      <filename>test.jpg</filename>
   </data>
</attachment>
```

Figure 27: An attachment element used for an image

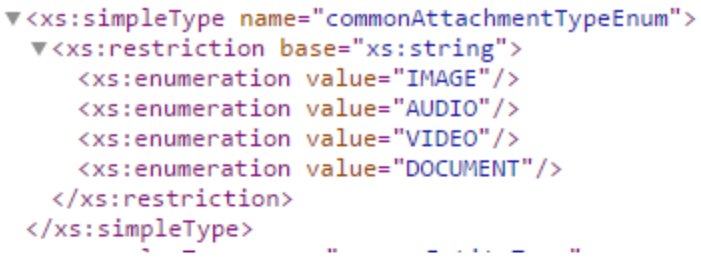
The attributes “type=IMAGE” and “observationType=CORRECTED” specify that the attachment is an image and it is a corrected image. If no “type”is specified (for observation attachments only), it will default to “IMAGE”.

The <data> element is a base64 representation of the image or file. This can be done with most frameworks programmatically, but to illustrate the concept you can use a tool such as https://www.base64-image.de/ to encode an image.

The following conditions will result in a validation error:

- The <data> element is not a valid `xs:base64Binary type`
- Type is set to anything other than IMAGE (observations only)
- observationType=”CORRECTED” and the observation is safe (corrected image should only be sent for unsafe observations)

The type of an attachment can also be other values, to designate different types:



```
▼<xs:simpleType name="commonAttachmentTypeEnum">
  ▼<xs:restriction base="xs:string">
     <xs:enumeration value="IMAGE"/>
     <xs:enumeration value="AUDIO"/>
     <xs:enumeration value="VIDEO"/>
     <xs:enumeration value="DOCUMENT"/>
   </xs:restriction>
 </xs:simpleType>
```

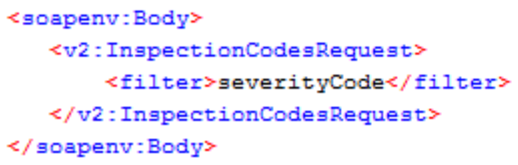
Figure 28: Atatchment types

The attribute observationType only applies to observation images, and can either be “CORRECTED” or “OBSERVED”.

For inserting attachments to incidents, use the same <attachment> element, but omit the observationType attribute.

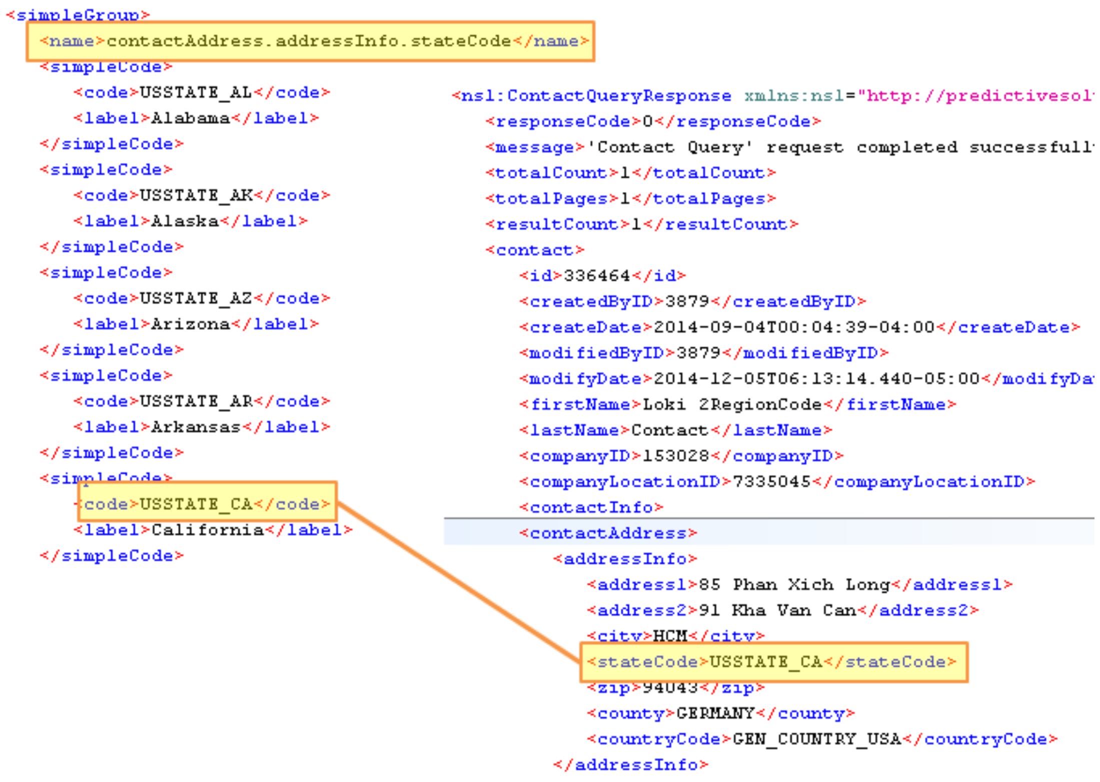
Lookup Codes

Lookup codes are nothing more than enumerations that are used in insert and update operations and returned in records when querying data. To determine which fields on a record use a lookup code value, execute the codes request for the specific record.



```
<soapenv:Body>
   <v2:InspectionCodesRequest>
       <filter>severityCode</filter>
   </v2:InspectionCodesRequest>
</soapenv:Body>
```

Figure 29: Codes request using a filter



```
<simpleGroup>
  <name>contactAddress.addressInfo.stateCode</name>
  <simpleCode>
     <code>USSTATE_AL</code>
     <label>Alabama</label>
  </simpleCode>
  <simpleCode>
     <code>USSTATE_AK</code>
     <label>Alaska</label>
  </simpleCode>
  <simpleCode>
     <code>USSTATE_AZ</code>
     <label>Arizona</label>
  </simpleCode>
  <simpleCode>
     <code>USSTATE_AR</code>
     <label>Arkansas</label>
  </simpleCode>
  <simpleCode>
     <code>USSTATE_CA</code>
     <label>California</label>
  </simpleCode>
```

```
<nsl:ContactQueryResponse xmlns:nsl="http://predictivesol
   <responseCode>0</responseCode>
   <message>'Contact Query' request completed successfull
   <totalCount>1</totalCount>
   <totalPages>1</totalPages>
   <resultCount>1</resultCount>
   <contact>
      <id>336464</id>
      <createdByID>3879</createdByID>
      <createDate>2014-09-04T00:04:39-04:00</createDate>
      <modifiedByID>3879</modifiedByID>
      <modifyDate>2014-12-05T06:13:14.440-05:00</modifyDa
      <firstName>Loki 2RegionCode</firstName>
      <lastName>Contact</lastName>
      <companyID>153028</companyID>
      <companyLocationID>7335045</companyLocationID>
      <contactInfo>
      <contactAddress>
         <addressInfo>
            <address1>85 Phan Xich Long</address1>
            <address2>91 Kha Van Can</address2>
            <city>HCM</city>
            <stateCode>USSTATE_CA</stateCode>
            <zip>94043</zip>
            <county>GERMANY</county>
            <countryCode>GEN_COUNTRY_USA</countryCode>
         </addressInfo>
```

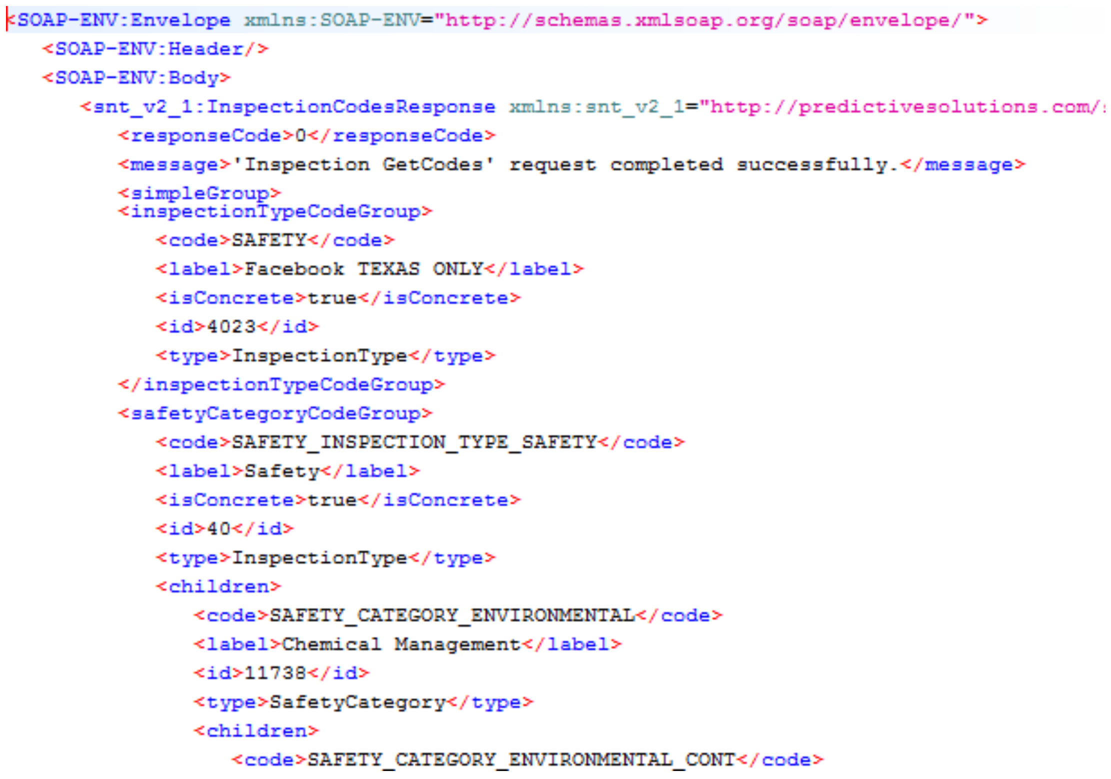
Figure 30: Example of SimpleCodeGroup and corresponding code in record

The <name> element of the <simpleGroup> will match a field in the corresponding record type. The <simpleGroup> element contains a list of <simpleCode> elements that specify the codes and labels used by the web service.

<label> is provided as ancillary information only and is never used in any web service request.

InspectionCodes Response

For the inspection service, the codes response contains the same <simpleGroup> but also additional information about inspection types, categories and custom data.



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <snt_v2_1:InspectionCodesResponse xmlns:snt_v2_1="http://predictivesolutions.com/:
         <responseCode>0</responseCode>
         <message>'Inspection GetCodes' request completed successfully.</message>
         <simpleGroup>
         <inspectionTypeCodeGroup>
            <code>SAFETY</code>
            <label>Facebook TEXAS ONLY</label>
            <isConcrete>true</isConcrete>
            <id>4023</id>
            <type>InspectionType</type>
         </inspectionTypeCodeGroup>
         <safetyCategoryCodeGroup>
            <code>SAFETY_INSPECTION_TYPE_SAFETY</code>
            <label>Safety</label>
            <isConcrete>true</isConcrete>
            <id>40</id>
            <type>InspectionType</type>
            <children>
               <code>SAFETY_CATEGORY_ENVIRONMENTAL</code>
               <label>Chemical Management</label>
               <id>11738</id>
               <type>SafetyCategory</type>
               <children>
                  <code>SAFETY_CATEGORY_ENVIRONMENTAL_CONT</code>
```

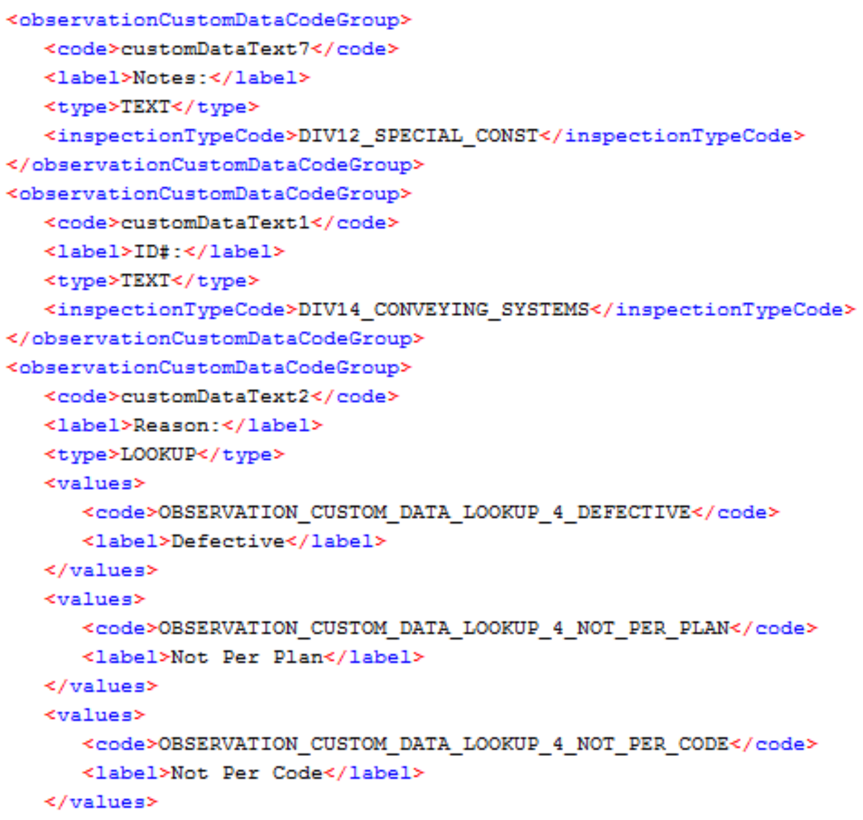
Figure 31: Codes response for Inspection Service

Custom Fields

Custom fields allow observations, inspections and incidents to be extended to contain customer specific requirements. For example, there may be a requirement to capture information about the weather during a safety observation. To achieve this, a custom field can be configured.

Observation Custom Fields

Custom fields are sent back in the Inspection Codes response in the <observationCustomDataCodeGroup> element.

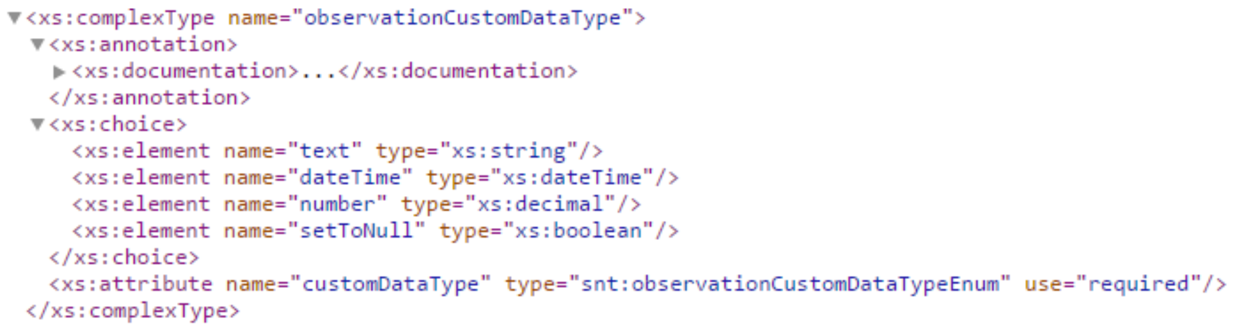


```
<observationCustomDataCodeGroup>
   <code>customDataText7</code>
   <label>Notes:</label>
   <type>TEXT</type>
   <inspectionTypeCode>DIV12_SPECIAL_CONST</inspectionTypeCode>
</observationCustomDataCodeGroup>
<observationCustomDataCodeGroup>
   <code>customDataText1</code>
   <label>ID#:</label>
   <type>TEXT</type>
   <inspectionTypeCode>DIV14_CONVEYING_SYSTEMS</inspectionTypeCode>
</observationCustomDataCodeGroup>
<observationCustomDataCodeGroup>
   <code>customDataText2</code>
   <label>Reason:</label>
   <type>LOOKUP</type>
   <values>
      <code>OBSERVATION_CUSTOM_DATA_LOOKUP_4_DEFECTIVE</code>
      <label>Defective</label>
   </values>
   <values>
      <code>OBSERVATION_CUSTOM_DATA_LOOKUP_4_NOT_PER_PLAN</code>
      <label>Not Per Plan</label>
   </values>
   <values>
      <code>OBSERVATION_CUSTOM_DATA_LOOKUP_4_NOT_PER_CODE</code>
      <label>Not Per Code</label>
   </values>
```

Figure 32: Inspection Service observation custom data

Note that custom fields are specific to an inspection type. In the example above, the "ID#" custom field belongs to "DIV4_CONVEYING_SYSTEMS" inspection type. This custom field could be defined for other inspection types, but it may have a different meaning.

When sending observation custom fields the observationCustomDataType is used.



```
▼<xs:complexType name="observationCustomDataType">
  ▼<xs:annotation>
    ▶<xs:documentation>...</xs:documentation>
   </xs:annotation>
  ▼<xs:choice>
     <xs:element name="text" type="xs:string"/>
     <xs:element name="dateTime" type="xs:dateTime"/>
     <xs:element name="number" type="xs:decimal"/>
     <xs:element name="setToNull" type="xs:boolean"/>
   </xs:choice>
   <xs:attribute name="customDataType" type="snt:observationCustomDataTypeEnum" use="required"/>
 </xs:complexType>
```

A few important notes about sending custom data:

- Only one of the elements from text, dateTime, number and setToNull can be used.
- setToNull specifies that the specific custom data should be cleared

- customDataType is specified with the observationCustomDataTypeEnum. These enumerations are sent back in the Codes response (if the customer is using custom data)
- custom data that is specified as type='LOOKUP' is still sent as text, LOOKUP just specifies that the text value must be one of the values specified in the codes response.

If using the custom field from the example above, the element in the observation would look like this:

```
<customData customDataType="customDataText2">
   <text>OBSERVATION_CUSTOM_DATA_LOOKUP_4_DEFECTIVE</text>
</customData>
```

Inspection Custom Fields

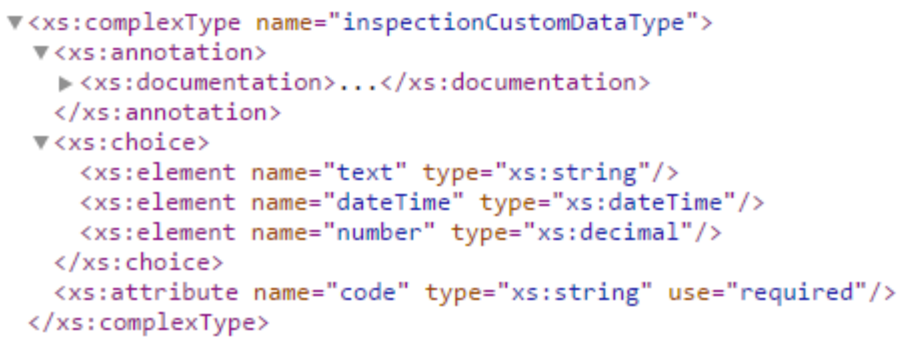
Custom data is sent back in the Inspection Codes response in the <inspectionCustomDataCodeGroup> element.

```
<inspectionCustomDataCodeGroup>
   <code>CONSTRUCTION_PHASE</code>
   <label>Phase:</label>
   <type>LOOKUP</type>
   <values>
      <code>CONSTRUCTION_PHASE_PREP_MEETING</code>
      <label>Prep Meeting</label>
   </values>
   <values>
      <code>CONSTRUCTION_PHASE_INITIAL_INSPECTION</code>
      <label>Initial Inspection</label>
   </values>
   <values>
      <code>CONSTRUCTION_PHASE_FOLLOW_UP_INSPECTION</code>
      <label>Follow-up Inspection</label>
   </values>
   <values>
      <code>CONSTRUCTION_PHASE_PUNCHLIST</code>
      <label>Punchlist</label>
   </values>
   <inspectionTypeCode>SAFETY_INSPECTION_TYPE_DIV6_WOOD_PLASTICS_INSPECTION</inspectionTypeCode>
</inspectionCustomDataCodeGroup>
```

Figure 33: Inspection Service inspection custom data

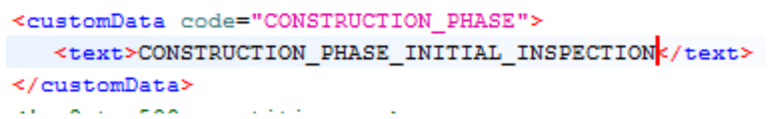
Note that custom fields are specific to an inspection type. In the example above, the "Phase:" custom field belongs to "SAFETY_INSPECTION_TYPE_DIV6_WOOD_PLASTICS_INSPECTION" inspection type. This custom field could be defined for other inspection types, but it may have a different meaning.

When sending inspection custom fields the inspectionCustomDataType is used.



```
▼<xs:complexType name="inspectionCustomDataType">
  ▼<xs:annotation>
    ▶<xs:documentation>...</xs:documentation>
   </xs:annotation>
  ▼<xs:choice>
     <xs:element name="text" type="xs:string"/>
     <xs:element name="dateTime" type="xs:dateTime"/>
     <xs:element name="number" type="xs:decimal"/>
   </xs:choice>
   <xs:attribute name="code" type="xs:string" use="required"/>
 </xs:complexType>
```

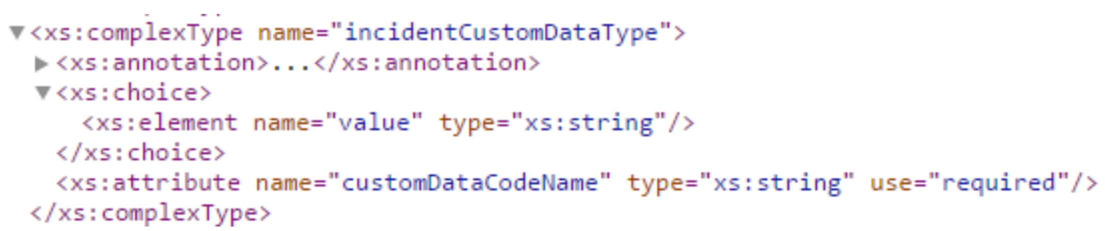
If using the custom field from the example above, the element in the inspection would look like this:



```
<customData code="CONSTRUCTION_PHASE">
   <text>CONSTRUCTION_PHASE_INITIAL_INSPECTION</text>
</customData>
```

Incident Custom Fields

Incident custom fields use the same process as Inspection Custom Fields above.



```
▼<xs:complexType name="incidentCustomDataType">
  ▶<xs:annotation>...</xs:annotation>
  ▼<xs:choice>
     <xs:element name="value" type="xs:string"/>
   </xs:choice>
   <xs:attribute name="customDataCodeName" type="xs:string" use="required"/>
 </xs:complexType>
```

Chapter 5: Data Types

Specific field types are not detailed in this document. Please reference the WSDL for specific record data type information.

XSD Data Types

Data types prefixed with the namespace "xs" are standard schema datatypes. The following are used by the SafetyNet Web Service:

- xs:anyURI
- xs:base64Binary
- xs:boolean
- xs:date[5] - specified in the following format "YYYY-MM-DD" where:
    - YYYY indicates the year
    - MM indicates the month
    - DD indicates the day
- xs:dateTime[5] - specified in the following form "YYYY-MM-DDThh:mm:ss.fff[+/-}offset" where:
    - YYYY indicates the year
    - MM indicates the month
    - DD indicates the day
    - T indicates the start of the required time section
    - hh indicates the hour
    - mm indicates the minute
    - ss indicates the second
    - fff indicates a fraction of a second
    - offset represents the timezone offset "Z" can be used to indicate Zulu time.
- xs:decimal
- xs:float
- xs:int
- xs:string

[5] Refer to http://www.w3schools.com/schema/schema_dtypes_date.asp for more information on date and time formats used in the SafetyNet Web Service.

SafetyNet Data Types

In addition to the standard XSD data types, the web service also uses several custom data types. These types are prefixed with the namespace "snt-common":

- snt-common:dateTimeType
- snt-common:idType – this is an extension if xs:int that only allows values of 1 or greater. It is used for all id fields
- snt-common:percentType – this is an extension of xs:int that only allows values from o to 100 (inclusive)
- snt-common:phoneType – an extension of xs:string that has a maximum of 30 characters
- snt-common:string80Type – an extension of xs:string that has a maximum of 80 characters snt-common:string255Type – an extension of xs:string that has a maximum of 255 characters
- snt-common:string8000Type – an extension of xs:string that has a maximum of 8000 characters
- snt-common:usernameType – an extension of xs:string that has a maximum of 50 characters and forces a valid username to characters and numbers

Enumerations

Enumerations are special data types that provide a list of values to be used for a field. They are simply an extension of xs:string, but fields that use an enumeration are limited to the values defined in the enumeration. Providing a value that is not in the enumeration will result in a validation error. The following enumerations are used:

- stateCodeEnum – a list of valid state codes used in addresses
- countryCodeEnum – a list of valid country codes used in addresses
- notificationStatusCodeEnum – a list of valid values for the notification status code
  - CONTACT_NOTIFICATION_STATUS_ALL
  - CONTACT_NOTIFICATION_STATUS_CUSTOM
  - CONTACT_NOTIFICATION_STATUS_DEFAULT
  - CONTACT_NOTIFICATION_STATUS_NONE
- safeEnum – a list of observation types used for querying
- courtesyTitleCodeEnum – a list of valid courtesy codes used for contacts
- suffixCodeEnum – a list of valid suffixes used for contacts

SafetyNet Record Fields

Most records contain the following common fields unless otherwise noted.

Field	Description	Type	Required
id	The unique identifier used when referencing the record in SafetyNet. This id must be set for an update and deleted operation.	snt-common:idType	update delete
createdByID	The id of the user that created the record. It is provided as a read only field; when inserting or	snt-common:idType	

	updating this field is ignored.		
createDate	The date and time that the record was created. It is provided as a read only field; when inserting or updating this field is ignored.	datetime	
modifiedByID	The id of the user that last modified the record. It is provided as a read only field; when inserting or updating this field is ignored.	snt-common:idType	
modifyDate	The date and time that the record was last modified. When updating and deleting, this field compared to the most recent version of the record's modifyDate in the database to prevent concurrency issues, and therefore must be included. See ***Chapter 2: Updating Data*** for more information.	datetime	update delete

Appendix A: FAQs

General

Which Request Should I Use?

Use the following table to determine which request to use based on your usage:

If you…	…then use this request:
…want to create a new record	Insert
…want to update an existing record	Update
…want to retrieve the full records that match your criteria	Query Request
…just want the total number of records in the database that match your criteria (not full records)	Count
…just want the ids of all records in the database that match your criteria (not full records)	GetID
…want the full record of an existing record	FindByID
… want to update a record and need the most current version	FindByID
…want to delete a record	Delete
…need to get a list of valid codes to use for inserting or updating an record	LookupCodes
…want to retrieve the company locations for a single company	CompanyAssociationQuery
…want to retrieve the project teams and segments for a single project	ProjectAssociationQuery
…want to create a new company location, project team or project segment	AssociationInsert
…want to update a company location, project team or project segment	AssociationUpdate
…want to delete a company location, project team or project segment	AssociationDelete
…want to know what fields are required for a specific record before I execute an insert or update operation	Fields (ConfigService only)

What programming languages are supported to call SafetyNet Web Services?

The SafetyNet web service can be used by any framework that supports SOAP.

Updating

I want to update records in SafetyNet. Can I send only the fields that I want to update? What will happen to existing data?

Missing fields or fields with an empty value are treated the same by the web service, the field will be cleared in the database. **Sending update requests with missing fields can lead to loss of data.** See Updating Data for more information.

Why does the modifyDate of a record have to match the modifyDate in the database?
This ensures that the record that is being updated or deleted is the most current version. When the modifyDate in the request does not match the modifyDate for that record in the database, an error is returned to the user stating such. The FindByID operation should then be invoked to get the most current version of the record so the user can update that version.

How do I send an image with an observation?
See ***Images and Attachments.***

Which version of the web service should I use?
Each version of a specific service is cumulative, that is each subsequent version provides the same functionality as the version before. Unless there reasons to use an older version, developers should use the most current version of a specific service to ensure that they have the latest functionality.

Appendix B: Troubleshooting

Before contact support with web service issues, there are a number of troubleshooting steps that can be taken to eliminate some common problems with the web service integration.

- Check to ensure that the request is making it outside of your organization's network. Can you access the WSDL page at http://ws.predictivesolutions.com/service? Is the endpoint URL correct?
- Can you log into SafetyNet with the same username and password?
- If the request is making it to the endpoint, what is the HTTP status code?
- If the status code is 200, then what is the response code from SafetyNet? See ***Response Codes*** for more information about specific codes.
- Is there a validation error? The message and details elements of the response will provide more details on validation issues. See ***Validation*** for more information.
- Try sending the same request with a tool such as Advanced Rest Client or SoapUI. Many times there is a problem with the client project and these tools can eliminate this possibility.

Appendix C: Contacting Support

After troubleshooting the issue with the steps in Appendix B, contact support and please include the following information for support:

- The service, operation and version being used (ex. Inspection Service Query v2_1)
- The framework/language you are using to access SafetyNet web services
- The response from the server
    - If HTTP 200, include the response code, message and details
    - If not HTTP 200, include the response body and specific HTTP code
- The endpoint URL
- The zone ID/URL and username
- The troubleshooting steps and results you have already tried

Note that our support does not resolve client proxy issues, project setup issues or write code for clients. However, we can help you troubleshoot issues if we are given enough information from above.

Appendix D: Web Service Versions

Some services support additional operations. The table below highlights each record type, version and available operations. Highlighted yellow operations denote new functionality with that specific version.

Service	version 1.1	version 2	version 2.1	version 2.2
ActionItem	*Not Implemented in this version*	*Not Implemented in this version*	Insert Update Delete Query Count GetIDs FindByID Codes	*Not Implemented in this version*
Company	Insert Update Delete Query Count GetIDs FindByID Codes AssociationInsert AssociationUpdate AssociationDelete AssociationQuery	*Not Implemented in this version*	*Not Implemented in this version*	*Not Implemented in this version*
Contact	Insert Update Delete Query Count GetIDs FindByID Codes	*Not Implemented in this version*	*Not Implemented in this version*	*Not Implemented in this version*
Config	ConfigFields	*Not Implemented in this version*	*Not Implemented in this version*	*Not Implemented in this version*
Hierarchy	*Not Implemented in this version*	*Not Implemented in this version*	Insert Update Delete	*Not Implemented in this version*

			Query Count GetIDs FindByID Codes	
Incident	Insert Update Delete Query Count GetIDs FindByID Codes	* Insert * Update Delete Query Count GetIDs * FindByID Codes * Added support for new attachmentType schema	Insert Update Delete Query Count GetIDs FindByID Codes * MonthlyData (added)	*Not Implemented in this version*
Inspection	Insert Update Delete Query Count GetIDs FindByID Codes	Insert Update Delete Query Count GetIDs FindByID Codes	* Insert * Update Delete * Query Count GetIDs * FindByID Codes * Added support for multiple inspectors	*Not Implemented in this version*
Observation	Insert Update Delete Query Count GetIDs FindByID Codes	* Insert * Update Delete Query Count GetIDs * FindByID Codes * Added support for new attachmentType schema	Insert Update Delete Query Count GetIDs FindByID Codes	*Not Implemented in this version*
Project	Insert Update Delete	Insert Update Delete	Insert Update Delete	*Not Implemented in this version*

	Query Count GetIDs FindByID Codes AssociationInsert AssociationUpdate AssociationDelete AssociationQuery	Query Count GetIDs FindByID Codes AssociationInsert AssociationUpdate AssociationDelete AssociationQuery	Query Count GetIDs FindByID Codes AssociationInsert AssociationUpdate AssociationDelete AssociationQuery * ManHour (added)	
User	Insert Update Delete Query Count GetIDs FindByID Codes	*Not Implemented in this version*	*Not Implemented in this version*	

: