IRWIN

Integration Specification

Incidents API v7.1

Prepared By: IRWIN Core Team

Last Updated: 12/20/2021

Contents

1 Introduction

Integrated Reporting of Wildfire Information (IRWIN) is a Wildland Fire Information and technology (WFIT) affiliated investment intended to enable an "end-to-end" reporting capability. IRWIN provides data exchange capabilities between existing applications used to manage data related to wildland fire incidents and resources. IRWIN services are focused on the goals of reducing redundant data entry, identifying authoritative data sources, and improving the consistency, accuracy, and availability of operational data. By interconnecting systems, new and updated information is automatically available to the different interagency systems and to a dashboard to provide queries and reports. This capability supports a number of needs and provides benefits throughout the wildland fire community, including:

1. Allow consistent reporting of data
2. Reduce the duplicate entry of data
3. Identify authoritative sources of data
4. Speed access to data located in diverse source systems
5. Increase data accuracy, and
6. Increase the availability of data

To facilitate this, IRWIN provides a common data exchange capability across all participating functional areas for capturing, reporting, and sharing event/incident information. It is an objective for IRWIN to facilitate data integration services among systems to support near real-time availability of new and updated information to the relevant interagency systems. This is primarily accomplished by integrating these systems through the IRWIN Application Programming Interface (API): a RESTful web API providing a common method to exchange wildland fire data.

1.1 Purpose and Audience

The Incident Integration Specification introduces and expands upon those topics necessary to begin data exchange through the IRWIN Incidents API. A formal discovery process is required to obtain an authentication credential, which allows access to the IRWIN Incidents API. This document is not a replacement for that process. In addition, IRWIN provides a separate API for data exchange of resource information.

The IRWIN Community is comprised of the IRWIN Core Team and IRWIN Extended Teams. The IRWIN Core Team is responsible for developing and supporting the technical integration based on requirements provided by the Wildland Fire Community. The IRWIN Core team is comprised of technical developers, data architects, business leads and implementation leads.

IRWIN Extended Teams represent the technical and businesspersons who support a system that exchanges data with other systems through the IRWIN Integration Service.

This document is intended for extended teams and particularly their system developers responsible for modifying their application for data exchange within the IRWIN Incident integration services.

1.1.1 Associated Documents

IRWIN Data Mapping Workbook

A workbook containing sheets for the IRWIN data element details and Authoritative Data Source (ADS) matrix. This google excel link is located in the library of the IRWIN Extended Team neighborhood of the Wildland Fire Lessons Learned site.

http://www.wildfirelessons.net

For access to the Wildland Fire Lessons Learned site, please contact Jaymee Fojtik, IRWIN Project Manager.

Data Exchange Environment User Guide

A guide intended to assist the community in understanding the data exchange environment as it is facilitated by IRWIN.

For access, an account on the Wildland Fire Lessons Learned Center is required.
http://www.wildfirelessons.net/...

1.2 Communication Network

1.2.1 IRWIN Apps

A collection of applications that enable users and extended team developers to support the data exchange process.

Accessing IRWIN Apps

Access to IRWIN Apps is granted using a GeoPlatform account (https://geoplatform.maps.arcgis.com). Authorization to use each app is explicitly granted using the Groups concept provided by the GeoPlatform.

Observer

Observer is a tool for discovering IRWIN incidents and resources and understanding the data exchange transactions that have occurred. Observer is available for all three IRWIN environments, and can be used to interact with both **root** and **next** versions of the IRWIN APIs.

- TEST: https://irwint.doi.gov/observer
- TEST/next: https://irwint.doi.gov/observer?v=next
- OAT: https://irwinoat.doi.gov/observer
- OAT/next: https://irwinoat.doi.gov/observer?v=next

- Production: https://irwin.doi.gov/observer

1.2.2 IRWIN Website

Public facing site providing information regarding IRWIN.

https://www.forestsandrangelands.gov/WFIT/applications/IRWIN/

1.2.3 IRWIN Project Wildland Fire Lessons Learned Community

Wildland Fire Lessons Learned "Public" exposure. Any user subscribed to the Wildland Fire Lessons Learned Site will have access to information regarding IRWIN. This is not intended to be technical, but is meant to educate those who are familiar with wildland fire.

http://wildfirelessons.net

1.2.4 IRWIN Extended Teams Wildland Fire Lessons Learned Community

Wildland Fire Lessons Learned "Subscription" only. Extended Teams are provided access to detailed information about the IRWIN project. Content includes final documents, release notes, integration specifications, and other communications pertinent to the business and developers of systems interacting with IRWIN.

For access to this community, please contact Brandon Green, IRWIN Project Manager.

http://wildfirelessons.net

1.3 Points of Contact

Chuck Wamack – IRWIN DOI Business Lead

dwamack@ios.doi.gov

208.334.6190

Brandon Green - IRWIN Project Manager

Brandon_Green@ios.doi.gov

410.303.3307

2 Conceptual Architecture

The IRWIN Incident API (Application Programmer Interface) is designed to broker common wildland fire INCIDENT data across various applications. This RESTful API exposes standard Add, Query, and Update utility operations, allowing integrated systems to share operational data. Although the API is customized, it follows standard extension guidelines of the underlying ArcGIS Server software. These custom operations:

- Validate data standards
- Enforce updates on an element-by-element basis only by authenticated systems
- Provide operations specific to the business needs of the wildland fire community

The API's role is to provide the ability for many disparate systems to create and edit incident information, or retrieve updated incident data on demand. With the understanding that these external systems leverage different core technologies, languages, platforms, are in varying lifecycle stages, or have different business rules, the API provides a common, flexible approach to integration yet enforces NWCG accepted standards and business workflows.

The workflow for incident creation, updates, and conflict resolution can be accessed by clicking here. This diagram provides a reference for external system business and technical persons for designing and testing the IRWIN integration interface. The "yellow" colored blocks depict actions taken by the IRWIN API when an external system adds, updates or queries incidents. The color "purple" in the conflict resolution table indicates values that are set by the external system.

3 Environments

IRWIN has three API environments: TEST, Operational Acceptance Testing (OAT), and Production (PROD). During any release, the release package is promoted from TEST to OAT to PROD. Each promotion only occurs after appropriate testing and acceptance. Each Extended System is given credentials to authenticate to each environment.

Within the TEST and OAT environments, there is a **next folder**. The software exposed in TEST/next and OAT/next is considered under-development. The software at the root is considered stable and is identical to the API on PROD. The following table describes each environment's intended purpose:

	TEST	OAT	Production
root	Extended Systems testing against released software. https://irwint.doi.gov/arcgis/rest/services	Extended Systems testing & QA against released software. https://irwinoat.doi.gov/arcgis/rest/services	Extended Systems using IRWIN API as an integration service. https://irwin.doi.gov/arcgis/rest/services
next	IRWIN Core Team testing against under-development software. https://irwint.doi.gov/arcgis/rest/services/next	Extended Systems testing against under-development software. https://irwinoat.doi.gov/arcgis/rest/services/next	

For more information about the release workflow across these environments, please reference the Release Management Plan.

3.1 Accessing Root v Next APIs

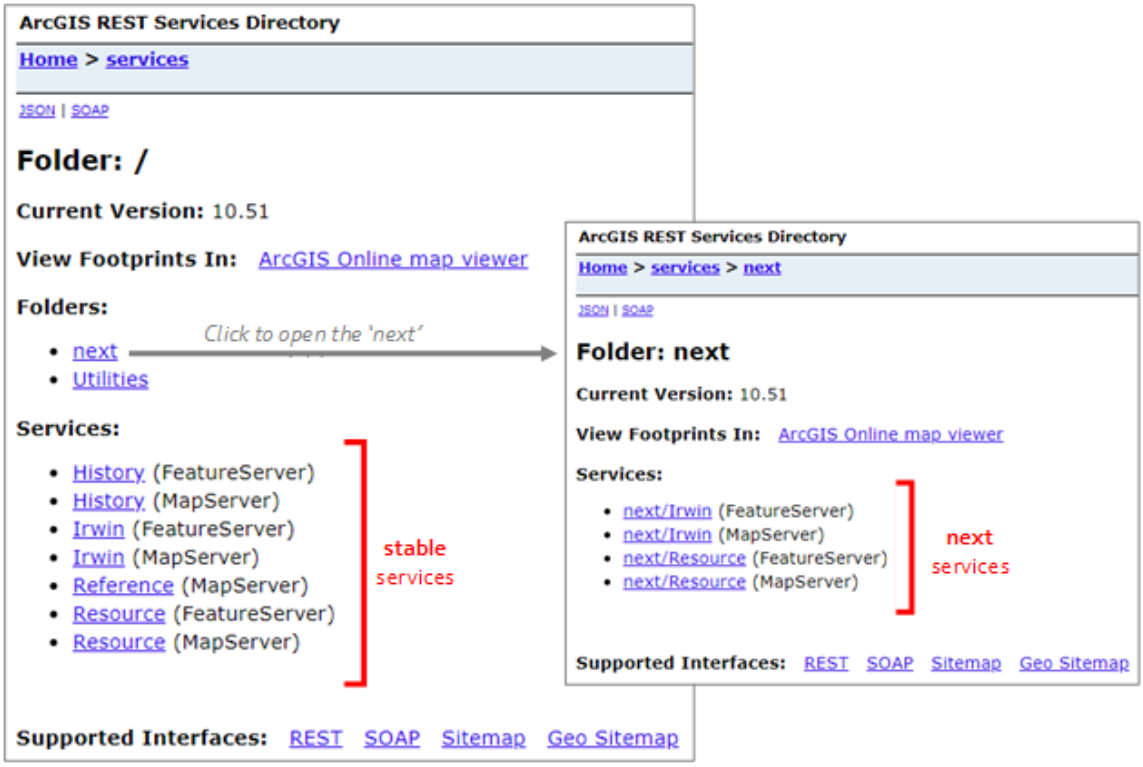
The root and next services are accessed through IRWIN's ArcGIS REST Services Directory, which varies by IRWIN environment:

Environment	ArcGIS REST Services Directory URL
TEST	https://irwint.doi.gov/arcgis/rest/services
OAT	https://irwinoat.doi.gov/arcgis/rest/services
Production	https://irwin.doi.gov/arcgis/rest/services

The root services can be accessed through the Services portion of the ArcGIS REST Services Directory. The Services portion of the ArcGIS REST Services Directory is accessible on all IRWIN environments.

The TEST/next and OAT/next services can be accessed by selecting 'next' under the Folders portion of the ArcGIS REST Services Directory. The 'next' folder is only accessible on IRWIN's TEST and OAT environments.

The figure below illustrates how to access both root and next services. These screens are for example purposes only, so the version number and layer/table list may be different when accessing real time.



ArcGIS API REST Endpoints Example Screen

3.2 Checking the API Version

To validate or check the API version to which you are connecting, use the value "IRWIN_API_Version" which is a property that can be found by reading the JSON response of either the root or any layer of the feature service.

4 Approach to Integration

Integration with IRWIN's Incident API involves analyzing the extended team system's user workflows to understand where their users create new incidents (ADD), read (QUERY), and edit or invalidate existing incidents (UPDATE). Each of these actions are "Integration Points" where the partner system may be adjusted to include calls to IRWIN web services.

> NOTE: IRWIN does not push data to connected systems, but rather allows connected systems to publish and consume data via services.

This analysis is known as the "Discovery Process". Over the course of this process, the primary focus is to understand alignment with common workflows to discover Integration Points. The workflows are diagrammed and then expanded to analyze how the integration with IRWIN might occur, as well as if additional requirements need to be analyzed with the Core team.

The outcome of the discovery process is a mutual understanding between the IRWIN Core and Extended teams regarding how to integrate. The following key questions drive a standard discovery process:

- What is the application's architecture and overall design?
- Does the application currently communicate over HTTPS?
- Does the system create new incidents? If so, what is the workflow?
- Does the system update existing incidents? If so, what is the workflow?
- What data elements does the system require to minimally define an incident?
- Does the application store incident data or require local processing?
- Does the system share or ingest information with other applications via data export, reports, or APIs?

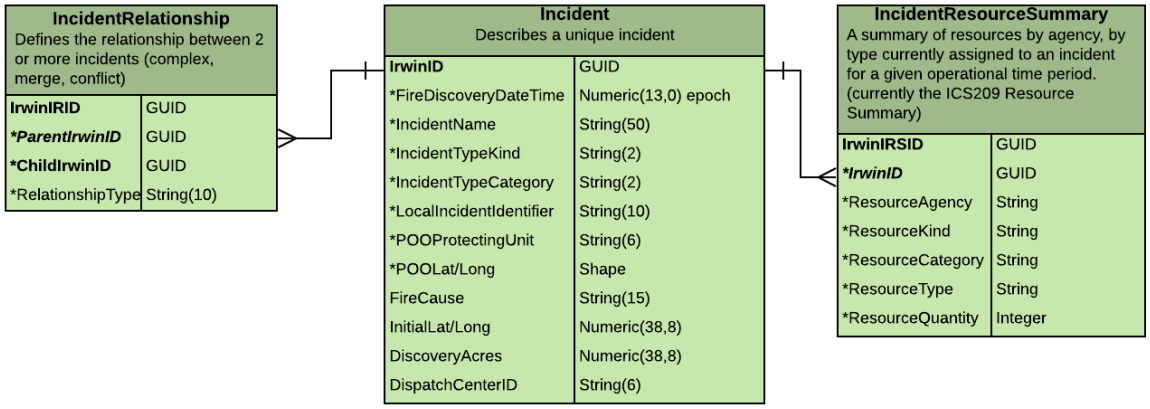
Following the Discovery Process, the system is provided credentials to TEST and OAT environments to begin their development against the IRWIN API. External systems using the API will require a level of integration testing, facilitated by the IRWIN business leads, before being issued credentials to the Production environment. This integration testing occurs between January and March each year.

Besides the specific integration points mentioned above, applications will need to maintain synchronization with IRWIN in order to acquire data updates from other participating systems. This may be accomplished by creating a process to continuously poll IRWIN at predefined intervals (seconds or minutes), or as users access the incident. Synchronization also includes taking action on specific "patterns" within an incident's data, which signal specific actions to occur, such as the deprecation of an incident because of duplication, transfer of ownership, or request for a fire code. In this manner, all systems should maintain a high degree of data integrity, allowing for more efficient data integration across applications.

5 Development Considerations

This section provides guidance to developers in understanding the main areas of coding for interacting with the Incidents API to exchange data. The incident service contains several feature service data layers, allowing RESTful interaction via exposed web operations. The figure below depicts the Incident Feature Server layer and related tables.

Incident Feature Layer and Related Tables



* Required fields.
NOTE: Not all fields are listed. Referenct the IRWIN Data Mapping Worksheet for details regarding each layer/table.

- Incident Layer
  - Describes an individual incident by its data element such as Unique Incident Identifier, Fire Discovery Date &Time, Incident Name, Point of Origin Latitude/Longitude and Fire Cause. There are over 125 Incident data elements, only 13 of which are required by a CAD to create an incident. Reference the IRWIN Data Mapping Worksheet, Incidents tab for details on every data element.

- Incident Relationships Table
  - Defines relationships between two or more incidents such as Complexes, Merges, Supporting and PotentialConflicts.

- Incident Resource Summary Table
  - A summary of resources by agency, by type currently assigned to an incident for a given operational time period (currently the ICS209 Resource Summary).

5.1 Authentication and Authorization

All integrated systems are provided a system level account to authenticate with the IRWIN Incident API. Systems will authenticate by acquiring a short-lived token string via the GenerateToken operation and adding its value to a token parameter when making any succeeding web calls to IRWIN. As part of the GenerateToken response, the token's expiration time is provided. Once the token's expiration time is met, the integrated system needs to request a new token.

> NOTE: The maximum token lifetime that may be requested in IRWIN is 60 minutes. A token should not be requested more than twice per hour. Best practice is requesting once every hour.

When generating a token, a parameter named 'client' is supplied. There are several options for this parameter as described in the online documentation.
It is recommended that systems use the 'referer' client as the supplied parameter. This option avoids issues such as IPs that may change between requests of getting the token and subsequent calls that use the token, and is a more stable option in environments where IPs may not always remain the same.
To implement this, when the token is requested, the client value is supplied as 'referer', and a value is supplied for the optional 'referer' parameter. This value for 'referer' can be any value, but will need to be supplied on subsequent calls that use the token and the two values must match.

Sample input:
username='yoursystem',
password='yourpassword',
client='referer',
referer='YOUR REFERER VALUE',
expiration=60,
f=json

On subsequent calls that use this token, the token value must be supplied, and the REFERER header value in any HTTP requests must match the value for 'referer' provided in the token request.

Documentation for the GenerateToken operation can be found at:

http://resources.arcgis.com/en/help/arcgis-rest-api/index.html#/Generate_Token/02r3000000ts000000/

Once a credential system connects to IRWIN, access to individual API operations and data elements is based on authorization roles. Each integrated system is placed into a role defined during the discovery process. For example, all read only systems have an "IRWINREAD" role.

Integrated systems that will write are described as "CAD" and "non-CAD" and have IRWINCAD and IRWINREADWRITE roles, respectively. The federal FireCode application is assigned the role of IRWINFIRECODE, this role is specifically designed for this application only.

For a full list of available roles specific to incidents, reference the table below.

Incident API Role	Detail
IRWINREAD	Role for systems that only read data from IRWIN. Grants access to read only operations: • `Query Incident, Incident Relationship, Incident Resource Summary` • `Resource and related information`
IRWINREADWRITE	Role for non-CAD systems, allowing read and write actions to IRWIN. Grants access to the following operations and enforces READWRITE required fields on Add or Update: • `Query Incident, Incident Relationship, Incident Resource Summary` • `Update Incident, Incident Relationship` • `Add Incident, Incident Relationship, Incident Resource Summary` With this role, the minimum required data elements for adding or updating an incident are: • FireDiscoveryDateTime • IncidentName • IncidentTypeCategory • IncidentTypeKind • LocalIncidentIdentifier • POOProtectingUnit • Shape (geometry object representing Latitude/Longitude at the incident's point of origin) – only required for Fire (FI) IncidentTypeKind

IRWINCAD	Role for CAD systems, allowing read and write actions to IRWIN. Grants access to the following operations and enforces CAD required fields on Add or Update: • `Query Incident, Incident Relationship, Incident Resource Summary` • `Update Incident, Incident Relationship, Incident Resource Summary` • `Add Incident, Incident Relationship, Incident Resource Summary` With this role, the minimum required data elements for adding or updating an incident are: • FireDiscoveryDateTime • IncidentName • IncidentTypeCategory • IncidentTypeKind • LocalIncidentIdentifier • POOProtectingUnit • Shape (geometry object representing Latitude/Longitude at the incident's point of origin) – only required for Fire (FI) or Fire Management Action (FM) IncidentTypeKind • FireCause • InitialLatitude • InitialLongitude • DiscoveryAcres • DispatchCenterID
IRWINFIRECODE	Role specifically for the FireCode system, allowing read and write data to IRWIN. Grants access to read and write operations: • `Query Incident, Incident Relationship` • `Update Incident, Incident Relationship` • `Add Incident, Incident Relationship` Allows FireCode to update the `FireCode` and `isFireCodeRequested` data elements. With this role, the minimum required data element for adding or updating an incident are the same as those for IRWINREADWRITE role.
IRWINFIREREPORTING	Role for fire reporting systems, allowing read and write actions to IRWIN. Grants access to the following operations and enforces Fire Reporting required fields on Add or Update: • `Query Incident, Incident Relationship, Incident Resource Summary` • `Update Incident, Incident Relationship` • `Add Incident, Incident Relationship, Incident Resource Summary` With this role, the minimum required data elements for adding or updating an incident are: • FireDiscoveryDateTime • IncidentName • IncidentTypeCategory • IncidentTypeKind • LocalIncidentIdentifier • POOProtectingUnit (if null on add or update, IRWIN will provide the value based on intersection of Jurisdictional Unit layer and POO Latitude/Longitude(as indicated by the x,y coordinates of the geometry shape object for the incident). • Shape (geometry object representing Latitude/Longitude at the incident's point of origin) – only required for Fire (FI) or Fire Management Action (FM) IncidentTypeKind

5.2 Key Data Concepts

For both reading and updating incidents, there are a few data elements that are fundamental to understan

ding the IRWIN incident data. Please reference the IRWIN Data Mapping Workbook for more detailed information on these data elements.

- IsValid - Indicates whether the incident is valid within IRWIN. Valid Incidents are records from the Dispatch Center having primary responsibility for that fire. Invalid incidents are a result of duplicates or invalid entries. When reading or updating incident records, filtering for IsValid = 0 should be taken into account as appropriate for the intended results. These records will also be marked for deletion by IRWIN.

- IsQuarantined - Indicates whether an incident of IncidentTypeKind FI (Fire) is potentially conflicting with another incident of IncidentTypeKind FI (Fire) based on IRWIN conflict detection rules. When submitting incidents, use this field to detect if the incident has been flagged as a potential duplicate. Quarantined incidents will not be visible to any other integrated systems. If a submitted incident has been flagged in quarantine, your system will need to present conflict resolution options to a user. See Section 6.1 for more details.
- FireOutDateTime - when setting the FireOutDateTime to a value, also set the ADSPermissionState to "FIREREPORTING".
- ADSPermissionState - Every system with write access to the API is part of the Authoritative Data Source (ADS) hierarchy. There are 4 permission states – DEFAULT, ICS209, FIREREPORTING and CERTIFIED. The field ADSPermissionState indicates the permission hierarchy that is currently being applied when a system utilizes the Update Incident operation. For every data element, each system is assigned a number that determines the priority that system has in updating a particular data element. If the element was last modified by a system with a higher value, then the data cannot be updated by a system with a lower ADS value. Once a record is in the "CERTIFIED" permission state, all data elements related to a final fire report can no longer be updated. Only systems with the role of "IRWINFIREREPORTING" can change the permission state back to DEFAULT, ICS209 or FIREREPORTING if the record should need to be updated. When a Fire Out date is updated from null by an external system, the incident should be set to the FIREREPORTING state by the system that sets the FireOutDateTime value. See Section 5.6 for more details.

- FireOutDateTime - When a Fire Out date is updated from null by an external system, the incident should be set to the FIREREPORTING state by the system that sets the FireOutDateTime value.

- IncidentTypeKind - A general, high-level code and description of the types of incidents and planned events to which the interagency wildland fire community responds. IRWIN uses the NWCG standard allowing many types of incidents including Fire and Natural Disasters.
- UniqueFireIdentifier – The UniqueFireIdentifier (UFI) is populated by Irwin by concatenating the year from the FireDiscoveryDateTime, the POOProtectingUnit and the LocalIncidentIdentifier. If Irwin finds the UFI is not unique to Irwin, the incident record will be rejected.
- IRWIN text fields do NOT allow ">" or "<", as those values could cause problems when ingested by other systems. For example ">>" can be interpreted as HTML by other systems, resulting in an error.

5.3 Authoritative Data Source (ADS)

The Authoritative Data Source (ADS) permission matrix defines a write system's permission to edit an element based on the incident's update history. By establishing a hierarchy for each write system, the ADS establishes a precedence of "authoritative" data. Higher precedence is given to more authoritative systems, on an element-by-element basis. In order for a particular system to update an element, that system must have equal or higher precedence than the last system to update it. The ADS exists to maintain data integrity as multiple systems begin to share information about a particular incident.

There are FOUR separate matrices to account for various stages of a wildland fire incident: DEFAULT, ICS209, FIREREPORTING and CERTIFIED. When an incident is first created, the incident adsPermissionState is set to "DEFAULT". In this configuration, the incident is primarily managed by a CAD system, thus CAD systems have the highest priority on almost every data element.

As an incident evolves, its management may require changing this hierarchy to allow the ICS209 system priority. In this scenario the CAD system "passes" ADS priority to ICS209 via updating the incident and setting its adsPermissionState to ICS209. This action transfers higher precedence of select elements to ICS209, allowing that system the ability to update fields it could not before. Other read/write system's hierarchy also changes with the ICS209 permission state. The CAD system may re-acquire precedence by adjusting the adsPermissionState back to Default. When a Fire Out date is entered, the incident should be set to the FIREREPORTING state by the system that sets the FireOutDateTime value. This state allows fire reporting applications to begin drafting a final fire report by setting their ADS values to the CAD level. Fire reporting applications have the ability to set the adsPermissionState to Certified essentially locking the data elements that comprise a final fire report record from further updates.

5.4 Reading Incidents

Reading incident data is accomplished through the ArcGIS REST QueryFeatures service layer operation referencing the incident feature layer number (0). This generic read operation allows for a wide variety of spatial and SQL where clause queries to be executed against the underlying data, returning an array of matched features. Query will accept IrwinIDs, UniqueFireIdentifiers, or any other search criteria in the form of a where clause, as well as specify which fields to return.

As part of the SOI (Server Object Interceptor), the API includes enhancements to the COTS (Commercial off-the-shelf) **Query** operation making custom request parameters available:

- includeADSStatus
- includeResources
- includeRelationships
- includeLastSyncDateTime

These parameters can be specified in the request parameters (ex.: "?includeADSStatus=true&includeResources=true", etc.) and then the associated data will be included in the response.

Reference Section 8, API Operations, for more details on custom parameters.

Documentation for Query can be found at:

http://resources.arcgis.com/en/help/arcgis-rest-api/02r3/02r3000000r1000000.htm

5.4.1 Maintaining Synchronization

Synchronization can be accomplished by periodically "polling" IRWIN using the ArcGIS REST **Query** operation referencing the Incident feature layer index number. The criteria included in the query's "where clause" can be written based on how the system wants to maintain synchronization.

Continuous Polling

Systems which store and process incident data locally may periodically poll IRWIN to acquire updated incident information. Using the Query API operation, systems will request updates between a start and end date/time on which the incident was modified or created – essentially requesting blocks of time. Updates returned are the current state of each incident that has been created or updated since the start date/time. This standard long-polling technique will require the application to maintain a fairly high level of synchronization in order to preserve cross-system integrity and maintain a smaller request payload.

Using Query, there are two common ways to implement this pattern:

1. Allow IRWIN to provide inputs for syncing: The most common way to accomplish sync is to provide an original value for the modifiedOnDateTime parameter to be used for the Query where clause and query for all incidents after this date. In the query response, IRWIN provides the next modifiedOnDateTime value as the nextSyncDateTime property. Use the nextSyncDateTime as the value to poll for in the subsequent call. This is the best way to keep up to date if the integrated system intends to maintain and process IRWIN updates locally. Note that the modifedOnDateTime is set when the incident is first created, so examining dates modified will also capture newly created incidents.

2. Request distinct blocks of time: Request distinct blocks of time by setting an upper and lower limit for the modifiedOnDateTime parameters in the Query where clause. This allows the client to fully control the block of time, and as long as blocks are contiguous, no updates will be missed.

Lazy Load Updates

Systems may choose to load IRWIN updates upon user accessing a particular incident. This pattern is appropriate only for those systems that originate all new incidents within their purview (such as a CAD) or have previously established context with an incident via an IrwinId. With an IrwinId or UniqueFireIdentifier in hand, the system may use the Query operation and supply the IrwinId(s) or UniqueFireIdentifier(s) as part of the where clause to acquire the latest state for one or more specific incidents.

5.5 Incident Creation

Incidents can be created using the ArcGIS REST AddFeatures operation. This generic create operation allows for individual or batch features to be created against the underlying data layer. AddFeatures will accept one or more standard feature objects, which express the Incident(s) the user wishes to create. Depending on the integrated system's authorization role, it is required to express a minimum number of data elements to successfully create the incident. In addition, all incidents that are IncidentTypeKind of FI (Fire) or FM (Fire Management) will require a latitude/longitude location expressed in the feature geometry object. Reference the *IRWIN Data Mapping Workbook* for details regarding required fields, data types, valid values and validation rules.

All submitted data elements are run against validation in order to enforce data standards. A successful creation will result in a response indicating success and the incident payload (i.e., IrwinId, and values of auto-calculated data elements) for the integrated system to act on. If the AddFeatures operation fails validation, an error response object is returned.

In addition, IRWIN will determine if the incident being submitted is potentially in conflict with an incident that already exists. The Incident Relationships Types section describes the workflow and dataflow for conflict detection and resolution.

Documentation for AddFeatures can be found at:

http://resources.arcgis.com/en/help/arcgis-rest-api/02r3/02r30000010m000000.htm

5.6 Incident Updates

Incidents are updated in IRWIN using the ArcGIS REST UpdateFeatures operation. This generic update operation allows for individual or batch features to be updated against the underlying data layer. UpdateFeatures will accept one or more standard feature objects, which express the Incident(s) the user wishes to update.

Documentation for UpdateFeatures can be found at:

http://resources.arcgis.com/en/help/arcgis-rest-api/02r3/02r3000000zt000000.htm

5.6.1 Authoritative Data Source (ADS) Permission Matrix

Although all write systems are capable of updating incidents, each data element is controlled by a hierarchical permission matrix that defines the Authoritative Data Sources (ADS). This matrix determines which integrated systems have precedence to update specific data elements. Any system with write permission can set a value for a specific element if it's null. The update to a specific element may be denied if another integrated system is more authoritative for the data element. If the ADS denies update privileges to a data element, it does not constitute failure for the entire update request, only the update of that particular data element is "skipped" as part of the request. A list of data elements not updated because of the ADS are listed in the response object.

All updated data elements are run against validation to enforce data standards. A successful update will result in a response indicating success, the incident payload that includes any auto-calculated fields based on updated data, and any ADS skipped elements. If the update fails validation, an error response object is returned. Reference the *IRWIN Data Mapping Workbook* for details regarding required fields, data types, ADS permission values, valid values and validation rules.

If a value is set back to *null* by an authoritative system, then the ADS permission is also reset allowing any other system to update the value as if it was being entered for the first time.

5.7 Auto-Generated Values in IRWIN

The IRWIN API has a series of auto-generated data elements, many are accompanied by additional logic. These auto-generated elements are spelled out below:

1. The IRWIN automatically sets the following default values when an Incident is created:
    a. ADSPermissionState = DEFAULT
    b. IncidentTypeKind = FI
    c. WFDSSDecisionStatus = No Decision
    d. IsDispatchComplete = 0 (false)
    e. HasFatalities = 0 (false)
    f. HasInjuries = 0 (false)
    g. SHAPE has a default spatial reference wkid 4269 (NAD 83)

2. IRWIN automatically sets the following fields when an Incident is created:
    a. CreatedOnDateTime = (now in UTC)
    b. CreatedBySystem = userid of system submitting the incident
    c. IrwinID = GUID randomly generated

3. IRWIN automatically sets the following fields when any data element in an Incident is updated:
    a. ModifiedOnDateTime = (now in UTC)
    b. ModifiedBySystem = (now in UTC)

4. IRWIN spatially derives the following fields based on the SHAPE (Point of Origin) supplied when an Incident is created:
    a. POOState
    b. POOCounty
    c. POOFips
    d. GACC
    e. POOPredictiveServiceAreaID
    f. POOJurisdictionalUnit
    g. POODispatchCenterID

- If DispatchCenterID is not supplied in an AddFeatures request, IRWIN populates the DispatchCenterID field with the geospatially derived POODispatchCenterID.

- For the IRWINFIREREPORTING role, POOProtectingUnit is not required on an AddFeatures request. If the POOProtectingUnit is not supplied on an AddFeatures request, IRWIN populates the POOProtectingUnit with the geospatially derived POOJurisdictionalUnit.

- On AddFeatures, POOProtectingAgency is derived from the POOProtectingUnit.
- On AddFeatures, POOJurisdictionalAgency is derived from the POOJurisdictionalUnit.
- On AddFeatures, UniqueFireIdentifier is derived as a concatenation of the following:
    1. YYYY from the FireDiscoveryDateTime
    2. POOProtectingUnit
    3. LocalIncidentIdentifier
- On an AddFeature and UpdateFeatures, if a user doesn't provide a value for the following values, IRWIN sets those not supplied to 0:
    1. FireStrategyConfinePercent
    2. FireStrategyFullSuppPercent
    3. FireStrategyMonitorPercent
    4. FireStrategyPointZonePercent

    Note: If any of these fields has a value, IRWIN does NOT reset the value to 0 when nulled. In this case, the system must update the value to 0, not null.
- If not supplied, IsFireCodeRequested is set to 0 (false) on AddFeatures.
- If a value for FireCode is supplied, IRWIN sets IsFireCodeRequested to 0 (false) on AddFeatures & UpdateFeatures.
- Conflict detection will potentially set IsQuarantined to 1 (true). If an incident is not flagged as in conflict, IRWIN sets IsQuarantined to 0 (false).

    Note: Users cannot supply values for IsQuarantined on AddFeatures. If it is supplied, IRWIN ignores the value.
- If IsQuarantined = 1 (true), IRWIN creates a conflict relationship:
    1. ChildIrwinID set to the IrwinID of the newly added incident
    2. ParentIrwinID set to the IrwinID of the existing incident
    3. IsActive set to 1 (true)
    4. RelationshipType set to "PotentialConflict"
    5. ReportedCreatedOnDateTime set to now()
    6. ReportedExpiredOnDateTime set to null

    Note: The newly added incident may be in conflict with multiple incidents; in this case, multiple relationship records are created for each conflicted incident.
- If IsValid is set to 0 (false), IRWIN expires all relationships related to the incident (i.e. the incident's IrwinID is contained within the ChildIrwinID or in the ParentIrwinID of the

Incident_Relationships table).

- When an incident relationship is updated to RelationshipType = 'ProvidingResponseTo' IRWIN automatically updates the IncidentTypeCategory of the child incident in the relationship to an 'OR'.

- If updating the ADSPermissionState of an incident that is the 'parent' in a relationship, IRWIN updates the ADSPermissionState of any 'child' incidents in those relationships. This does not apply to ADSPermissionState of CERTIFIED, this will not cascade to 'child' incidents.

6 Incident Relationship Types

Relationships between incidents can be established through the IRWIN API. Four types of relationships can be defined between one or more incidents:

- Complex: A relationship between a Complex (`IncidentTypeCategory` of `CX`) and two or more Wildfires (`IncidentTypeCategory` of `WF`) formed when it is decided the identified Wildfires should be managed together.

- PotentialConflict: A relationship between two or more Fires (`IncidentTypeKind` of `FI`) formed when the identified records are reported as potential duplicates of one another. The Conflict Resolution workflow will be executed to identify if either are duplicates and establish the valid incident (identified as the parent).

- Merge: A relationship between two or more Wildfires (`IncidentTypeCategory` of `WF`) formed when they physically burn into one another.

- ProvidingResponseTo: Describes the relationship between a "child" incident (created by a dispatch center) that is providing resources in response to the same incident to which a different dispatch has protection responsibility and has also created the "parent" incident.

Relationships are created using the ArcGIS REST AddFeatures operation on the Incident Relationships feature layer. Each record in this relationship table indicates a relationship between TWO records in the Incident layer. In the relationship, the two incidents' unique IrwinIDs are recorded. One incident's IrwinID will be recorded in the ParentIrwinID and the other in the ChildIrwinID data element.

Each relationship in the Incident_Relationships table contains additional fields relevant to the relationship:

- ***IsActive*** - a boolean indicator identifying if the relationship is active (1) or expired (0)
- ***RelationshipType*** - indicates the type of relationship (PotentialConflict, ProvidingResponseTo, Complex, or Merge)
- ***ReportedCreatedOnDateTime*** - UTC time (epoch format) when the relationship was reported - this might vary from the auto-generated CreatedOnDateTime indicating when the relationship was created within IRWIN
- ***ReportedExpiredOnDateTime*** - UTC time (epoch format) when the relationship was expired - this field is auto-generated by IRWIN when IsActive is set to false (0)

Each relationship type is created through a specific order of operations. Refer to the IRWIN Data Mapping Worksheet & the Incident Workflows v6 tab of the LucidChart Diagrams for information.

6.1 PotentialConflict

Incident_Relationships of type 'PotentialConflict' are only created by IRWIN. When an incident is added, that incident is subject to a series of quarantining rules to determine if that incident is potentially in conflict with other incident(s).

6.1.1 Potential Conflict Detection

Each add request to the Incident layer triggers conflict detection. The incident being added is compared to all other incidents (potential parents). Potential parents must be:

IsValid = 1 *(true)*,

IsQuarantined = 0 *(false)*,

IncidentTypeKind = FI~~, and~~

~~IncidentTypeCategory ≠ CX or FA~~

The added incident is compared to each potential parent for the following criteria. All 5 conditions must be met to result in a potential conflict:

1. Both incidents must be of IncidentTypeKind 'FI'
2. ~~Both Incidents must not be IncidentTypeCategory 'CX' or 'FA'~~
3. The FireDiscoveryDateTime of the added incident must be within 6 hours (< 6 hours) of the potential parent incident
4. The Geometry (SHAPE@XY) of the added incident must be within ½ miles (<½ miles) of the potential parent incident
5. Both incidents must have either different values for DispatchCenterID OR different values for CreatedBySystem
    a. For example, Incident A DispatchCenterID = IDBOC and Incident B DispatchCenter = IDBOC, but A was created by Wildcad and B was created by Firecode, the two incidents would be flagged as potential conflicts.

By default, if the incident is not flagged as in potential conflict, IsQuarantined is set to 0 (false) and IsValid is set to 1 (true) by IRWIN.

If all conflict detection rules are met...

(1) an Incident_Relationship of type 'PotentialConflict' is created by IRWIN for each incident to which the child is found to be in conflict.

Child IrwinID	Parent IrwinID	Is Active	Relationship Type	Reported CreatedOn DateTime	Reported Expired On DateTime
IrwinID *of the newly added incident (the child)*	IrwinID *of the existing incident (the parent)*	1 (True)	PotentialConflict	now()	Null

(2) the newly added incident's attributes are updated to reflect IsQuarantined = 1 (True).

This child incident's record that is trying to be added will look like:

Is Quarantined	Is Valid	IncidentType Category
1 (True) *Automatically set by IRWIN where quarantining*	1 (True) *This value is un-changed from the originally submitted*	*Whatever B was submitted with ~~(must not be CX~~, but must be of IncidentTypeKind*

rules met	*incident*	*FI)*

6.1.2 Potential Conflict Resolution

The CreatedBySystem should present the conflict alongside the parent(s) for resolution. There are three scenarios for resolving conflict as described below. For the case where both records may have valid information for the incident and the data needs to be aligned between the parent & child incident, administrators within each system should communicate necessary changes to the participating systems so that the valid incident reflects the most accurate data.

Scenario 1 - Child Lost (is not a valid incident)

The Child CreatedBySystem should:

1) Set the child incident to IsValid = 0 (False)
2) Set the child incident to IsQuarantined = 0 (False)
3) Set the incident relationship IsActive = False (0) effectively expiring the relationship.
4) Set the ReportedOnReportedExpiredOnDateTime = now().

Note: IsQuarantined cannot be nulled.

Scenario 2 - Child Should be an OR (Out of Area Response)

The Child CreatedBySystem should:

(1) NOTE: Ensure all capability requests on the child incident are set to FulfillmentStatus = "Closed"
(2) Set the PotentialConflict incident relationship IsActive = 0 (False) & the ReportedExpiredOnDateTime = now().
(3) Create a new active relationship of type "ProvidingResponseTo" between the parent and the child.
(4) IRWIN will update the IncidentTypeKind to "FM" and IncidentTypeCategory to "OR"
(5) Set the child incident to IsQuarantined = 0 (False)
(6) Set the child incident Incident.isValid = 0 (False)

Note: IsQuarantined cannot be nulled.

Scenario 3 - Both the Parent and the Child Win (both are valid distinct incidents or the child is a False Alarm)

In this scenario, the child is still a different and valid incident from the parent. The Child CreatedBySystem should:

(1) Set the child incident to IsQuarantined = 0 (False).
(2) Set the incident relationship IsActive = False (0) effectively expiring the relationship.
(3) If the child incident is a False Alarm, update the IncidentTypeKind "FM" and IncidentTypeCategory to "FA".
(4) Set the ReportedOnReportedExpiredOnDateTime = now().

Note: IsQuarantined cannot be nulled.

6.2 Complex

6.2.1 Creating Complexes

To create a complex and then add incidents to that complex, follow the steps below:

(1) Create a complex by creating a record in the Incident layer of IncidentTypeKind = 'FM' and IncidentTypeCategory = 'CX'
- ★ An incident record CANNOT be updated to an IncidentTypeCategory = 'CX' - the incident record has to be created as a IncidentTypeCategory = 'CX'.
- ★ An incident record of IncidentTypeCategory = 'CX' cannot be updated to any other IncidentTypeCategory after it was initially added.
- ★ An incident record of IncidentTypeCategory = 'CX' must include "Complex" as a standalone word in the Incident name. This is not case-sensitive.

(2) Add wildfire incidents to the complex by creating record(s) in the IncidentRelationships layer with the following attributes:
- RelationshipType = 'Complex'
- ParentIrwinID = Incident IrwinID of the record with IncidentTypeKind = 'FM' and IncidentTypeCategory = 'CX'
- ChildIrwinID = Incident IrwinID of the record with IncidentTypeKind = 'FI' and IncidentTypeCategory = 'WF'
  - ★ If the complex is intended to relate more than one incident of IncidentTypeCategory = 'WF', create an Incident_Relationships record for each.
  - ★ Where RelationshipType = 'Complex', there is validation within IRWIN to ensure that the ParentIrwinID relates to an incident of IncidentTypeCategory = 'CX' and the ChildIrwinID related to an incident of IncidentTypeCategory = 'WF'.
  - ★ 209 Business Rule: The child incident being added to a complex must have a 209 state of Final or Null. In 209, the incident 209 can't be approved if it has been added to a complex during Initial or Update. (note: this rule is not enforced by IRWIN but should be adhered to by external systems)

6.2.2 Invalidating an Incident Complex

An incident of type complex may need to be removed and the incidents no longer associated with that complex. A complex is removed by setting the complex incident (of IncidentTypeCategory = 'CX') to IsValid = 0 (False).

When the complex record is set to invalid, IRWIN will set all records in the relationship table with ParentIrwinID equal to that complex's IrwinID to IsActive = 0 (False).

6.2.3 Removing Incidents from a Complex

To remove an incidents from a complex, invalidate the the relationship record in the IncidentRelationships table for the record where ChildIrwinID = [IrwinID of incident to be removed from complex]

- isActive = false.

6.2.4 Moving Incidents from One Complex to Another

Moving an incident from one complex to another complex is a 2-step process:

1. First, remove the incident from the complex they are currently in by following steps in 6.2.3.
2. Second, add the incident to the complex they are being moved to following the steps in 6.2.1 step 2.

6.3 Merge

6.3.1 Creating Merges

Incidents can be merged when they have grown together - one incident gets consumed by another incident. To merge 2 or more incidents Incident_Relationships of type 'Merge' are created by relating the records being merged as described below:

(1) Create record(s) in the IncidentRelationships layer with the following attributes:
- RelationshipType = 'Merge'
- ParentIrwinID = Incident IrwinID of the incident record that other incidents will be merged into.
- ChildIrwinID = Incident IrwinID of the record incident record being merged into the parent incident.

(2) Update the FireOutDateTime for the child record being merged.

(3) [IROC/CAD will define how resources on the child incident are dealt with and IRWIN can add the directions here if necessary]

6.3.2 Removing Merges

A merge is removed by setting the record in the Incident_Relationship table to IsActive = 0 (False).

7 Error Handling

The IRWIN API returns a variety of indicators and status codes detailing the success or failure of actions. Upon addFeatures or updateFeatures, a boolean "success" property is returned, indicating if the action was successful or not. If false, an error property is also returned which lists the error **code** (indicating the kind of error) and **description** (providing the actual error messages).

Note: If an element is prevented from being updated due to the system's ADS permission hierarchy, that element(s) will be listed in the skippedElements portion of the response.

Additional Documentation for error responses can be found at:

https://developers.arcgis.com/rest/services-reference/feature-service-error-codes.htm

IRWIN Exception Codes include:

Description	Code
Initialization Exception	8001
Configuration Exception	8002
Unauthorized Exception	8003
Validation Exception	8004
Json Geometry Exception	8005
Operation Failed Exception	8006

7.1 Validation Errors

If the request results in one or more validation errors, the response will include an "error" object with the "code" property specified as 8004. The "description" property of the error object will be an array of validation error objects. Each validation error that is relevant will be included as a separate object with one of the following codes and messages.

JSON Syntax:

```
{
  "objectId":  <objectId>, //int, objectId value of the updated/inserted feature
  "globalId":  <globalId>, //string, string globalId value of updated/inserted feature
  "success":  <true | false>, //boolean, false if edit was not applied
  "error": { //only returned if success is false
    "code":  <code>, //integer, error code
    "description": [ //array of validation error objects
      {
        "error": {
          "code":  <code>, //integer, error code
          "message":  <message>, //string, validation error message/description
          "conflictObjectId": <conflictObjectId> //integer, only returned if validation error is a "unique" conflict
        }
      }
    ]
  }
}
```

In the following tables, text highlighted in gray represents example values only; the actual text may vary based on the input and/or context.

Code	Example Message	What does it mean?	How to fix it?
101	value (This is wrong!) must be composed of alphanumeric,	The value may only contain letters, numbers, hyphens	Remove any characters from the value that are not letters,

	hyphen, or period characters.	(-), and/or periods (.)	numerical digits, hyphens, or periods.
103	value (X) must be an accepted value ([A|B|C]).	The value must be one of a defined list (or "domain").	Ensure the value you are passing matches one of the specified values in the domain values for this field. Note that case may be important.
105	Invalid type. Expected number.	The value must be a number; spaces, letters, or other non-numeric characters are not allowed. Periods and hyphens may be allowed if they are contextually relevant, such as for floating point or negative values.	Ensure the value is a number with no spaces.
	value (10) must not exceed 5.	The numerical value must be less than or equal to the stated maximum.	Lower the value to less than or equal to the maximum.
106	value (abcdefg) must not exceed 6 characters.	The value is too long.	Shorten the length of the value.
107	Invalid type. Expected number.	The value must be a number; spaces, letters, or other non-numeric characters are not allowed. Periods and hyphens may be allowed if they are contextually relevant, such as for floating point or negative values.	Ensure the value is a number with no spaces.
	value (1) must exceed 5.	The numerical value must be more than or equal to the stated minimum.	Raise the value to more than or equal to the minimum.

108	value (tooshort) must be at least 15 characters.	The value is too short.	Lengthen the value to be at least the minimum length; spaces are not valid padding. Typically this means left-padding the value with zeroes.
109	value is not nullable.	The value cannot be omitted on addFeatures or nullified on updateFeatures.	For addFeatures requests, you must include a non-null value. For updateFeatures, ensure the value is not being set to "null".
110	value contains disallowed characters.	The value contains characters that are not allowed for this field, such as spaces or special characters.	Check the value to ensure it contains only characters relevant to this field data.
111	a value is required for IrwinCAD systems.	(Systems with IrwinCAD role only) This value is required on addFeatures.	Ensure the value is not missing or null.
112	a value is required.	This value is required.	Ensure the value is not missing or null.
113	value (XXWRNG) must begin with a valid state code.	The first two characters of this string value must be a valid state abbreviation (or, in certain cases, "CA" or "MX").	Ensure the first two characters are a valid NWCG-standard state code (or "CA" or "MX", if the relevant data falls within Canada or Mexico accordingly)
114	value (123) must be type string.	The value must be a string, and cannot be passed as a JSON-specified type of boolean, number, array, or object.	Ensure the value is enclosed by quotes.

	value (3.14) must be type integer.	The value must be a whole number.	Ensure the numerical value is a whole number with no decimal.
	value (wrong) must be type float.	The value must be a number.	Ensure the value is a number.
	value (Sunday, 23 Feb 2019) must be type epoch datetime (long integer).	The value must be a valid datetime value in Unix-time (or "epoch"-time) format. See this link.	Ensure that the value is a datetime value, expressed as a whole number of milliseconds after 12:00 am, January 1, 1970. This will be a 13-digit number.
	value ([34.01,-117.34]) must be type geometry.	The value was not recognized as a correctly formatted JSON geometry. See this link.	Ensure the value is a correctly formed JSON object, with (at a minimum) a value for "x" and "y".
	value ({"lat":34.01,"lon":-117.34}) must be a correctly formed geometry object. See https://developers.arcgis.com/documentation/common-data-types/geometry-objects.htm#POINT	The value was not recognized as a correctly formatted JSON geometry. See this link.	Ensure the value is a correctly formed JSON object, with (at a minimum) a value for "x" and "y".
200	Unable to validate whether IncidentTypeCategory matches IncidentTypeKind because the value of 'IncidentTypeCategory' could not be parsed.	The Incident API couldn't read or parse the value specified for "IncidentTypeCategory".	Ensure the value of "IncidentTypeCategory" does not contain invalid characters and is an accepted value.

	Incident Type Category 'WF' does not match Incident Type Kind 'HZ'	The value specified for IncidentTypeCategory is not a known category for the value specified for IncidentTypeKind.	Change the value of either IncidentTypeCategory or IncidentTypeKind so that IncidentTypeCategory is a known category for kind. For example, an IncidentTypeCategory of "WF" must be an IncidentTypeKind of "FI".
	Error querying table "EventKindAndCat": Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.
201	Incident category cannot change while the incident is in a 'Complex' relationship.	It is not possible to change the value of IncidentTypeCategory of an incident that is in an active relationship (parent or child) of type "Complex" or "Merge".	Expire any active relationships this incident is a member of before attempting to change the value of IncidentTypeCategory.
	Error querying for IrwinID 069BA152-C519-4502-A560-3F72754FB862: Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.
	Error parsing IrwinID	The IRWIN API couldn't parse the specified ID value.	Ensure the IrwinID/ IrwinRSID is a valid GUID, specified as a string value with no leading/trailing braces.
202	total value (50) of fields FireStrategyMonitorPercent, FireStrategyConfinePercent,	The sum of the values for those fields is neither 0 nor 100.	Ensure that all 4 of the relevant fields either: 1. Equal 0, or

	FireStrategyPointZonePercent, FireStrategyFullSuppPercent does not equal any of (0,100)		2. Add up to 100
203	If FireCode is not null, a new FireCode cannot be requested.	Once a FireCode has been provided, it is not possible to set the value of IsFirecodeRequested to 1.	Remove the property "IsFirecodeRequested" from future update requests for this incident.
	FireCode cannot be changed once it is set.	FireCode cannot be changed once it is set.	Remove the property "FireCode" from future update requests for this incident.
205	value (069BA152-C519-4502-A560-3F72754FB862) must be an existing IrwinID.	The specified IrwinID was not found in the incident layer.	Check to make sure you are using the correct IrwinID.
	Error querying for IrwinID (069BA152-C519-4502-A560-3F72754FB862): Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.
206	value (069BA152-C519-4502-A560-3F72754FB862) must be an existing IrwinID.	The specified IrwinID was not found in the incident layer.	Check to make sure you are using the correct IrwinID.
	Error querying for IrwinID (069BA152-C519-4502-A560-3F72754FB862): Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.

	value (069BA152-C519-4502-A560-3F72754FB862) is not valid.	The specified IrwinID refers to an incident that contains IsValid = 0 (false).	Check to make sure you are using the correct IrwinID. Otherwise, you may need to update the relevant incident to set IsValid = 1.
207	Unable to validate 1552957074764 because the value of FireDiscoveryDateTime could not be parsed.	Validation for the specified field depends on the value of another field (in this case, FireDiscoveryDateTime), but that field could not be read correctly so validation for the original field could not be completed.	Fix the input for the dependent field (in this case, FireDiscoveryDateTime).
	value (1552957074429) must precede 730 days after FireDiscoveryDateTime (1532957074768).	The specified datetime value must not be after 730 days from the value of the referenced field (in this case, FireDiscoveryDateTime). For example, if FireDiscoveryDateTime is January 1, 2017, then the value must be on or before December 31, 2018.	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
	value (1552957074442) must precede FireDiscoveryDateTime (1562957074456).	The specified datetime value must be before the value of the referenced field (in this case, FireDiscoveryDateTime).	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
	value (1602957074965) must precede 730 days from now (1552957074750).	The specified datetime value must not be after 730 days from today.	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
208	Unable to validate 1552957074335 because	Validation for the specified field depends on the value of	Fix the input for the dependent field (in this

	the value of FireDiscoveryDateTime could not be parsed.	another field (in this case, FireDiscoveryDateTime), but that field could not be read correctly so validation for the original field could not be completed.	case, FireDiscoveryDateTime).
	value (1552957074321) must succeed 5 days before FireDiscoveryDateTime (1532957074352).	The specified datetime value must not be before 5 days prior to the value of the referenced field (in this case, FireDiscoveryDateTime). For example, if FireDiscoveryDateTime is January 15, 2017, then the value must be on or after January 10, 2017.	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
	value (1552957074431) must succeed FireDiscoveryDateTime (1532957074213).	The specified datetime value must not be before the value of the referenced field (in this case, FireDiscoveryDateTime).	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
	value (1552957074634) must succeed 5 days (1557957074836).	The specified datetime value must not be before 5 days prior to today.	Ensure the datetime value falls within the specified limit and is correctly encoded in UNIX time + milliseconds format.
209	a value is required for incident type kind FI category WF.	This value is required if the incident has IncidentTypeKind = 'FI' and IncidentTypeCategory = 'WF'	Provide a non-null value for the specified field.
210	Cannot determine parent IrwinID.	The IRWIN API couldn't find or parse the specified ParentIrwinID value. That is, a valid ParentIrwinID was not found in the request.	Ensure the ParentIrwinID is a valid GUID, specified as a string value with no leading/trailing braces.

	Cannot determine child IrwinID.	The IRWIN API couldn't find or parse the specified ChildIrwinID value. That is, a valid ChildIrwinID was not found in the request.	Ensure the ChildIrwinID is a valid GUID, specified as a string value with no leading/trailing braces.
	Error querying for IrwinIds (069BA152-C519-4502-A560-3F72754FB862, 069BA152-C519-4502-A560-3F72754FB863): Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.
	Parent incident type category must be 'CX' to create a relationship of type 'Complex'.	It is only possible to create an incident relationship with RelationshipType = 'Complex' if the parent incident has IncidentTypeCategory = 'CX'.	Ensure the ParentIrwinID refers to an incident of type category 'CX'.
	Child incident type category must be 'WF' to create a relationship of type 'Complex'.	It is only possible to create an incident relationship with RelationshipType = 'Complex' if the child incident has IncidentTypeCategory = 'WF'.	Ensure the ChildIrwinID refers to an incident of type category 'WF'.
	Parent incident type category must be 'WF' to create a relationship of type 'Merge'.	It is only possible to create an incident relationship with RelationshipType = 'Merge' if the parent incident has IncidentTypeCategory = 'WF'.	Ensure the ParentIrwinID refers to an incident of type category 'WF'.
	Child incident type category must be 'WF' to create a relationship of type 'Merge'.	It is only possible to create an incident relationship with RelationshipType =	Ensure the ChildIrwinID refers to an incident of type category 'WF'.

		'Merge' if the child incident has IncidentTypeCategory = 'WF'.	
211	value cannot be changed.	Once this value is set, it cannot be changed.	Remove this value from future updateFeature requests for this record.
	value cannot be changed to CX.	The value cannot be changed to the value identified in the error message (in this case, 'CX').	Change the value or remove it from the updateFeature request.
212	If value is CX, it cannot be changed.	Once this value is set to the value identified in the error message (in this case, 'CX'), it cannot be changed.	Remove this value from future updateFeature requests for this record.
213	Unable to validate whether value is required because the value of 'IncidentTypeKind' could not be parsed.	IncidentTypeKind was either not included in the request or could not be parsed correctly.	Ensure there is a valid value for IncidentTypeKind in the request.
	a value is required for incident type kind FI category WF.	If the incident has IncidentTypeKind = 'FI' and IncidentTypeCategory = 'WF', the specified value is required to be non-null.	Set a non-null value for the specified field.
800	The following IrwinIDs are already assigned to this UniqueFireIdentifier **2019-AZXYZ-LOCALID1**: 069BA152-C519-4502-A560-3F72754FB862	UniqueFireIdentifier consists of *{FireDiscoveryDateTimeYear}-{POOProtectingUnit}-{LocalIncidentIdentifier}*. If the combination of these three fields is identical to that of another incident, this validation – and the request – will fail.	Ensure the three values are unique in combination. Typically the LocalIncidentIdentifier may be the easiest to change to ensure a unique combination.

802	Point of Origin does not intersect a US county.	The specified geometry object refers to a location that is not found within the reference map of US counties.	Ensure the location is correct, and that the units are in the correct format of decimal degrees latitude and longitude.
810	~~Child and parent are already in a relationship of type Merge.~~	~~A new relationship cannot be made for the given child and parent because they are in a relationship already (in this case, of type 'Merge').~~	~~There are several options:~~ ~~• Ensure the ParentIrwinID and ChildIrwinID are correct~~ ~~• Cancel further addFeatures requests to create a new relationship~~ ~~• Expire the existing relationship, then try again~~
811	Child is already in a conflict relationship.	A new relationship cannot be made for the given child because it is in a conflict relationship already.	Clear the conflict according to prescribed conflict resolution procedures.
812	~~Child is already in a Merge relationship.~~	~~A new relationship cannot be made for the given child because it is in a relationship already (in this case, of type 'Merge'). Incidents can only be a child of one 'Merge' or 'Complex' relationship.~~	~~There are several options:~~ ~~• Ensure the ParentIrwinID and ChildIrwinID are correct~~ ~~• Cancel further addFeatures requests to create a new relationship~~ ~~• Expire the existing relationship, then try again~~
813	Complex Parent IncidentTypeCategory must be 'CX' and Complex Child	It is only possible to create an incident relationship with RelationshipType =	Ensure the ParentIrwinID and ChildIrwinID are correct and that the parent incident has IncidentTypeCategory =

	IncidentTypeCategory must be 'WF'	'Complex' if the parent incident has IncidentTypeCategory = 'CX' and the child incident has IncidentTypeCategory = 'WF'.	'CX' and the child incident has IncidentTypeCategory = 'WF'.
814	Merges may only occur between incidents with IncidentTypeCategory of WF	It is only possible to create an incident relationship with RelationshipType = 'Merge' if both the parent incident and the child incident have IncidentTypeCategory = 'WF'.	Ensure the ParentIrwinID and ChildIrwinID are correct and that both the parent incident and the child incident have IncidentTypeCategory = 'WF'.
900	IrwinID is invalid.	The IRWIN API couldn't find or parse the specified ID value. That is, the ID was not found in the request.	Ensure the IrwinID/ IrwinRSID is a valid GUID, specified as a string value with no leading/trailing braces.
901 -904	IrwinID (069BA152-C519-4502-A560-3F72754FB862) not found.	The IRWIN API was unable to find a record in the relevant table/layer with the specified ID.	Ensure the IrwinID/ IrwinRSID is a valid GUID and refers to an existing record in the relevant layer.
905	Error querying for IrwinID 069BA152-C519-4502-A560-3F72754FB862: Database error	Depending on the error, this may indicate a server failure of some kind.	Check your input data and try again. If this happens repeatedly, please report it to the IRWIN implementation team.

8 API Operations

The Incidents API operations can be accessed at the following link. The *SWAGGERhub* documentation provides an interactive platform for both understanding method specifications

and also testing interacting with the methods. Note that this documentation tool is connected to the IRWIN TEST environment only.

https://app.swaggerhub.com/apis-docs/IRWIN7/Irwin/6.0.0

https://app.swaggerhub.com/apis-docs/IRWIN7/Resource/6.0.0