POLICEBOX

PoliceBox

API Developer Reference

Document Control

1. Version Control

Title:	API Developer Reference
Document reference:	PB2019080203API
Document version:	1.0
Document status:	Released

Issue	Date	Author	Summary of change
0.1	1 July 2019	C Eccles	Initial Draft
1.0	3 August 2019	C Eccles	Initial release following review

2. Approval & Distribution

Approved By:	Simon Hall	Chris Eccles
Company Title:	ceo	cto
Date:	03 AUG 2019	03 AUG 2019

Approved Distribution:	• Coeus Staff (Job Role Dependent) • Partners and Stakeholders when needed, with NDA in place. • Customers e.g. where needed as evidence.

Contents

3. Purpose of this document

This document is intended for developers and solution architects. It describes the functionality of the PoliceBox API. The PoliceBox API provides RESTful-style web services to support a range of integration processes. This document summarises the functionality available and gives some guidance on using the API to provide a fully integrated tasking system. The initial focus of the API is the management of tasking; more functionality will be added over time.

PoliceBox also provides an SDK for integration. The API is generally most appropriate when processes are initiated from outside PoliceBox, and the SDK is most appropriate when processes are initiated from within PoliceBox.

Related documents:

- An overview of API security issues is covered in a separate document, 'PoliceBox API Security and Data Integrity Protections'.
- Integration using the SDK and PoliceBox Hybrid Server is described in the document 'PoliceBox Connector SDK Reference V01'.

3.1. API Reference Updates

As PoliceBox continues to evolve through its product development roadmap, the API will also continue to evolve. For partners and other stakeholders with a systems integration role, it is important to ensure that you have the latest version of this document. It will be made available for authorised download on the knowledgebase https://kb.policebox.com.

4. Background

PoliceBox provides a flexible mobile digital platform for carrying out tasks. A task is defined as a piece of work which results in the use or capture of information. Tasks contain various properties plus a collection of fields which can be used to record structured information. Tasks are designed and published in the drag-and-drop PoliceBox App Designer and may also be created directly through the API.

A mobile user may initiate their own tasks, or tasks can be sent to a mobile user via the API or via the hybrid integration server. A user may search multiple remote databases and use the results to populate tasks. Data may also be intelligently copied ('cloned') from any one task to corresponding fields in any other.

Often a user takes on a role defined by a 'resource name' or 'callsign'. The use of callsigns allow tasks to be assigned without having to know which particular person is on duty at a given time. The API supports both sending tasks to named users or sending tasks to callsigns / resource names. The terms 'callsign' and 'resource' or 'resource name' are used interchangeably in this document.

When a task is sent to a callsign, the task typically has two functions – to inform the associated user of the task details, and to provide a means for the user to record the outcome of that task. The outcome may be recorded by completing additional fields in the task, or by copying the task into one or more locally generated tasks as required by the circumstances.

The results are then sent back to the PoliceBox platform from where the information is exported to the customer's back end systems as needed. This can be achieved via the PoliceBox Hybrid Server, which monitors the completion of tasks and calls out to third party systems via integration connectors.

5. Differences between the PoliceBox API and the Hybrid server (SDK)

PoliceBox provides two integration pathways which may be used individually or together; an API and a 'Hybrid Server'. The primary difference between these two integration points is as follows:

- The PoliceBox Hybrid server, supported by an SDK, is a set of installed services which call out to external systems in response to events occurring within PoliceBox. The Hybrid server is used to export data collected in PoliceBox and to run database searches initiated by mobile end users. The Hybrid Server is typically installed on the customer's network. This gives access to the customer's back end systems as well as national systems as appropriate. Specific integrations are implemented via connectors. Connectors are small self-contained code modules which perform the necessary translation between PoliceBox software and the third-party system. An SDK is provided to help write the connectors.

- The PoliceBox API is an HTTPS-based web service allowing third party systems to call in to PoliceBox in response to external events. The API typically resides with the PoliceBox back-end as a scalable web service and is called from the customer's integration systems, which may be on-premise or in the cloud. The API is loosely coupled to the PoliceBox software providing a simple yet powerful interface. The API is typically used to pass new tasks from a command and control system to mobile end users.

For example, generating a new task and sending it to a user would typically be an API function as it is externally initiated, whereas exporting a completed task would typically be a Hybrid Server function as it is internally initiated. Performing a biometric search from a mobile device would be a Hybrid Server function.

A customer can integrate their back-office systems directly with the API (if supported) or via some middleware or standard integration platform. A standard integration platform can be particularly useful if it already provides ready-built integration with the customer's other back office systems.

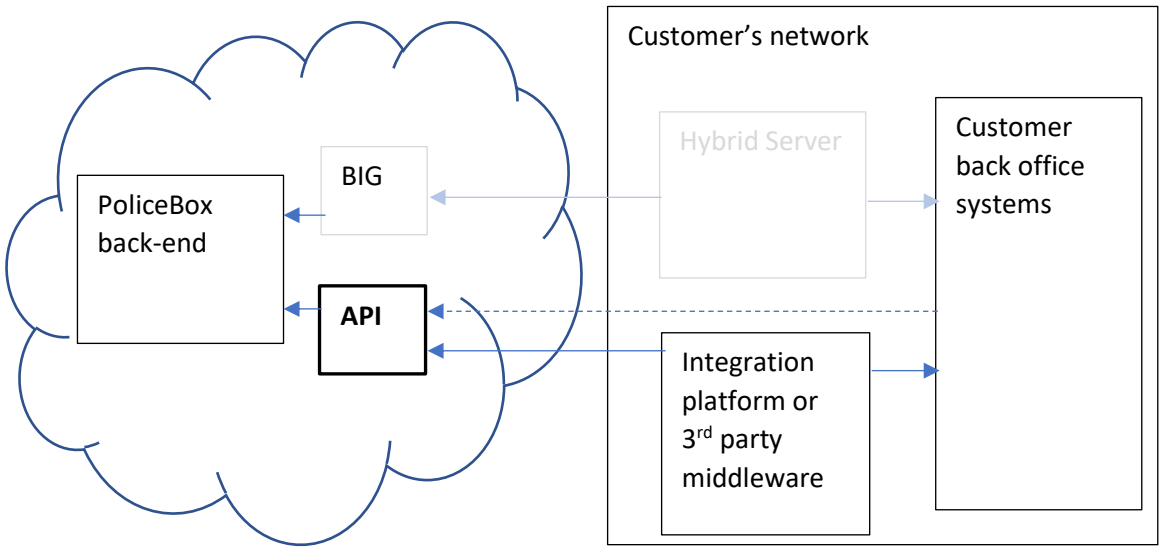
In the next release, completed tasks may also be processed via the API if desired, possibly eliminating the need for installation of the hybrid server in some circumstances. However, the Hybrid server provides deeper integration capabilities and core integration logic (such as automated retries etc) are provided out of the box.

5.1. Hybrid Server vs API comparison

Feature	Hybrid server	API
Requires installation on the customer's network	Yes	No
Type of installation	IaaS / VM	Normally SaaS
Primary Functionality	Exporting, Searching	Tasking, simple exporting
Scalability	Depends on customer's provision; size for peak loads	Highly scalable
Fault tolerance	Requires at least two servers with failover, two interconnected datacentres for georedundancy	Fault tolerance built in with geo-redundancy if cloud-based
Connection to third party systems	Calls out via connectors; PoliceBox connector needed for each target system	Called into via third party middleware or integration platform.
Mobile search support	Hosts RIG web service to provide searching against on-prem and national data sources	Does not provide searching.

Feature	Hybrid server	API
Coupling to PoliceBox software	Tight coupling gives powerful capabilities in connectors but may require connector updates for new software releases	Loose coupling – API more stable over time, providing access to most useful features but not the full software stack.

Figure 1: How the API and Hybrid Server sit in a typical integration architecture. Both approaches are optional or can be used together. The API is the focus of this document.



- BIG: Back-end Integration Gateway: An internal API used by the PoliceBox back-end.
- API: Customer-facing API for integration use, particularly for managing the sending of tasks to mobile users.
- Hybrid Server: Server(s) running on the customer's network which calls out to third party systems, for exporting collected data, performing mobile user database searches. Connector modules, written using an SDK, are used to provide connectivity with third party interfaces.

6. Typical tasking process

A customer will typically have some kind of system that generates tasks needing to be done. This may be manual or automatic.

When the need for a task arises, the next step is to work out who to give it to. This may be determined externally, but the PoliceBox platform provides functionality to support this process if required.

For mobile workers, the choice of assignment can be based on a range of criteria, such as:

- Proximity of the user to the task's location
- Whether the user is currently available to take on the task
- Whether the user has the necessary capabilities (e.g. qualifications, authority)
- Whether the resource the user is associated with has the necessary capabilities (e.g. hardware)

If we can answer these questions, we can arrive at a sorted shortlist of users to send the task to. Other custom considerations may also be applied externally by the systems integrator such as checking diaries for existing commitments or calculating drive times based on real time traffic conditions.

Having sent the task to the user, the user may accept or reject it. If the user rejects the task (giving a reason), the customer needs to find another user to send the task to if the task is still current.

If the user accepts the task, the user carries out the task, recording associated information and sending this back when complete.

If circumstances change, or if there was an error, the customer may rescind the task before it is complete. This alerts the user's device that the task is no longer current for that user and the task is abandoned.

If for any reason the user can't complete the task, it can be abandoned by the user (giving a reason) and sent back in an abandoned state.

When the completed task is received by the customer, it is then processed in conjunction with the customer's back office systems. This might result in the generation of one or more further tasks, in which case the process repeats.

An abandoned task can also be processed, resulting in possible follow-up such as the generation of a new task.

7. API general overview

The API is evolving over time and the range of functionality will grow. The API currently supports the following functions:

- Matching tasks to users or resources by location, status and capability
- Creating and sending tasks to users or resources, including embedded images
- Tracking active tasks in progress
- Getting the most recent location and status of logged-in users or resources
- Assign users to resource names (individually or in bulk)
- Define resource names (individually or in bulk)
- Define user capabilities (individually or in bulk)
- Define resource capabilities (individually or in bulk)

At the time of writing, the following features are due for the next release:

- Retrieving completed tasks for ongoing export, and reporting outcome
- Retrieving associated media files for ongoing export, and reporting outcome
- Situational awareness: current status and location of resources
- Metadata: anonymised task data to drive performance metrics

The API follows common RESTful principles. Actions use common HTTP verbs GET, POST, PUT, DELETE and PATCH (for bulk updates). All calls are via https / TLS1.2 or higher and require a system key and a strong system key in the URI, in addition to authentication credentials. The relevant values will be supplied to the customer as part of the API registration process.

The API will return an appropriate HTTP status code in response to a call. In the case of an error the body may contain additional information.

Parameters and Json properties are validated for length and value ranges. If a submitted item fails validation, Bad Request will be returned.

The API is Swagger/OpenAPI enabled, providing self-documentation and the potential to automatically create API clients.

To view the Swagger documentation, go to:
`[api address]/Swagger/ui/index`

To download the Swagger JSON document, go to
`[api address]/Swagger/docs/v1`

8. Details of API functionality

8.1. Get candidate resources/callsigns for a task

Before assigning a task, the system can provide a shortlist of callsigns most suited to the task based on their capabilities, current status and location. Resources, and the capabilities of users and resources may be defined via separate API calls.

8.1.1. Swagger documentation

`/Swagger/ui/index#!/AssignableResources/AssignableResources_Get`

8.1.2. Call details

`GET: api/syskey/strongsyskey/AssignableResources + query string`

8.1.3. Query parameters (all optional)

- maxMatches: The maximum number of matches to return (default = 10)
- targetLatitude: The latitude of the task's location (Wgs84 coordinates) (default = 0) Results are sorted by proximity.
- targetLongitude: The longitude of the task's location (Wgs84 coordinates) (default = 0) Results are sorted by proximity.
- includeAvailabilityCodes: CSV of up to 3 availability codes. A user must match one of them to be included. (Users set their current availability through their mobile app)
- excludeAvailabilityCodes: CSV of up to 3 availability codes: A user must not match any to be included.
- userCapabilities: CSV of up to 5 user capabilities. The user linked to the resource must match all specified capabilities to be included.
- resourceCapabilities: CSV of up to 5 resource capabilities. The resource must match all specified capabilities to be included.

8.1.4. Returned JSON data format

```
[
  {
    "DistanceMetres": 0,
    "Resource": "string",
    "Availability": "string",
    "AvailabilityDateTime": "2019-06-04T09:35:27.242Z",
    "LatitudeWgs84": 0,
    "LongitudeWgs84": 0,
    "LocationDateTime": "2019-06-04T09:35:27.242Z"
  }
]
```

The result returns a list of matching resources, sorted in order of increasing distance from the target location.

8.1.5. JSON properties details

- DistanceMetres: The direct-line distance of the resource from the target location in metres, based on last known position
- Resource: The name of the resource
- Availability: Indicates the resource's current availability code
- AvailabilityDateTime: The UTC time that the resource last reported its availability
- LatitudeWgs84: The last reported location of the resource using WGS84 coordinate system

- LongitudeWgs84: The last reported location of the resource using WGS84 coordinate system
- LocationDateTIme: The UTC time that the resource last reported its position.

The chosen resource can then be assigned the task via a separate API call.

The returned data may optionally be used to drive further business logic to determine the optimal resource to use for the task. (e.g. existing appointments, drive time based on travel conditions etc)

8.2. Get candidate users for a task

This call is identical to Get Candidate Resources, except user names are included in the return results. This can be useful if resource names are not used.

If user names are not explicitly required, use the resources version as this does not expose personal user names.

8.2.1. Swagger documentation

```
/Swagger/ui/index#!/AssignableUsers/AssignableUsers_Get
```

8.2.2. Call details

```
GET: api/syskey/strongsyskey/AssignableUsers + query string
```

8.2.3. Returned JSON

```
[
  {
    "DistanceMetres": 0,
    "User": "string",
    "Resource": "string",
    "Availability": "string",
    "AvailabilityDateTime": "2019-06-04T09:35:27.267Z",
    "LatitudeWgs84": 0,
    "LongitudeWgs84": 0,
    "LocationDateTime": "2019-06-04T09:35:27.267Z"
  }
]
```

8.2.4. JSON properties details

- DistanceMetres: The direct-line distance of the resource from the target location in metres, based on last known position
- User: User's login name
- Resource: The name of the resource
- Availability: Indicates the resource's current availability code
- AvailabilityDateTime: The UTC time that the resource last reported its availability
- LatitudeWgs84: The last reported location of the resource using WGS84 coordinate system
- LongitudeWgs84: The last reported location of the resource using WGS84 coordinate system
- LocationDateTime: The UTC time that the resource last reported its position.

8.3. Send a task to a user or resource/callsign

Json sent in the request body is used to define a task at a target location for a specified user or resource. The API will generate the task and send it to the user. Note that these tasks are generated independently of any task configuration used for self-tasking. To ensure task information clones across to locally generated tasks, ensure the field settings match equivalent fields in the configuration generated by the App Designer.

8.3.1. Swagger documentation

`/Swagger/ui/index#!/ActiveTaskDetails/ActiveTaskDetails_Post`

8.3.2. Call details:

`POST: api/syskey/strongsyskey/ActiveTaskDetails`

8.3.3. JSON structure (mandatory items are bold)

```
{
  “User”: “string”,        //either user or resource must be supplied.
  "Resource": "string",    //either user or resource must be supplied.
  "TaskDescription": "string",
  "Roles": "string",
  "TaskTitle": "string",
  "RefId": "string",
  "TaskName": "string",
  "TaskInits": "string",
  "Fields": [
    {
      "Prompt": "string",
      "Type": "string",
      "DictName": "string",
      "FieldName": "string",
      "IsReadOnly": true,
      "IsVisible": true,
      "RepeatingMax": 0,
      "Text": "string",
      "Code": "string",
      "Numeric": 0,
      "ImageType": "string",
      "ImageBase64": "string"
    }
  ],
  "ProtectiveMarking": "string",
  "RetentionDays": 0,
  "LatitudeWgs84": 0,
  "LongitudeWgs84": 0,
  "Address": "string",
  "AcceptExpiry": "2019-06-04T09:35:27.164Z",
  "TargetCommence": "2019-06-04T09:35:27.164Z"
  "TargetFinish": "2019-06-04T09:35:27.164Z",
  "TerminateExpiry": "2019-06-04T09:35:27.164Z",
  "IconPngBase64": "string",
  "TaskStateData": "string"
}
```

8.3.4. Minimal JSON structure

As most of the properties adopt sensible default values, the minimum essential JSON to specify a task is as follows:

```
{
  "Resource": "string",        // (or User)
  "TaskDescription": "string",
  "Roles": "string",
  "Fields": [
    {
      "Prompt": "string"
    }
  ]
}
```

If any of these essential properties are omitted a BadRequest will be returned.

Properties not specified will adopt default values. A field will default to the text datatype.

8.3.5. Complex tasks

To maintain a clean and simple API interface, tasks generated through the API do not support some of the more complex features that can be specified in the App Designer, such as SubForm functionality and conditional navigation logic within the task.

Where one or more complex forms need to be completed by the end user to complete the task, it is recommended that a simpler form is sent to the user containing information to initiate the task, this can then populate a more complex preconfigured task when received at the user's device.

8.3.6. JSON properties details

Item	Description	Constraints
User	The login name of the target user, e.g. name@domain.com. This does not need to be supplied if Resource is specified instead.	75 chars max. (Either User or Resource must be specified).
Resource	The resource name (aka callsign) of the target user. This can be specified in place of the User. Tasks are often assigned to resource names or callsigns instead of named individuals. The API supports bulk mapping of users to callsigns.	75 chars max. Either User or Resource must be specified. Mapping of users to resources is defined separately.
TaskTitle	The title of this particular task (e.g. "Attend disturbance in Main St").	80 chars max.
TaskDescription	A textual description of the task to be done (can contain more detail than the title)	2,000 chars max.
RefId	A unique string-based ID assigned to this task. If not specified, a GUID string is generated and returned in the response.	50 chars max.
TaskName	The name of this type of task, e.g. 'Assignment'	50 chars max
TaskInits	The initials of this type of task, e.g. 'ASGN'	10 chars max
Roles	A CSV list of user roles required for this task, as defined in Active Directory.	255 chars max. For user to access the task, at least one role must match a role in the user profile

Item	Description	Constraints
Fields	A collection of fields to be presented to the user. These may be prepopulated or empty awaiting the user to fill them in.	There must be at least one field.
Type	The field's datatype.	50 chars max. See notes below
DictName	The field's data dictionary name	100 chars max. To copy the field into other forms, this should match a configured item in the system's data dictionary, accessible through the App Designer application. If not supplied, a unique GUID string is generated.
FieldName	The name of this field	100 chars max. Usually same as DictName unless multiple fields using the same Dictionary item.
Prompt	The prompt shown to the user for this field.	4000 chars max (for signature field declarations), though normally <80 chars
IsReadOnly	If true, the user can't edit the contents of the field.	True or false
IsVisible	If false, the user can't see the field, but it is programmatically accessible for integration purposes	True or false (usually true)
RepeatingMax	The number of instances of the field allowed at run time.	1-100. Normally 1, unless the user is allowed to create more at run time (e.g. photos)
Text	The initial raw text value stored in the field.	1000 chars max. See notes on Type below.
Code	The initial code value stored in the field. (Not shown to user but may be useful for integration e.g. to store a key value)	50 chars max
ImageType	If the type is ImageEmbedded, specifies the type of image stored	If used, png or jpg
ImageBase64	If the type is ImageEmbedded, specifies the image data, encoded as Base64	If used, Base64 up to 100,000 chars max (<50,000 recommended)
ProtectiveMarking	An optional text string indicating the protection level of the data	50 chars max.
RetentionDays	The number of days the PoliceBox system will retain the completed form after it has been processed. If the form is configured to be exported, it won't be weeded until all assigned exports have successfully completed.	Zero or positive integer. There is an additional grace period of 24 hrs before the form is physically deleted
LatitudeWgs84	If known, the task's target location latitude, using the standard GPS WGS84 coordinate system	-90.0 to +90.0. If unspecified, set to 0.0
LongitudeWgs84	If known, the task's target location longitude, using the standard GPS WGS84 coordinate system	-180.0 to 180.0, If unspecified, set to 0.0
Address	If known, the task's target address	255 chars max
AcceptExpiry	The date and time by which the user must accept or reject the task (after which the task may be rescinded)	UTC date time string in ISO8601 format, i.e. `yyyy-mm-ddThh:mm:ssZ`
TargetCommence	The target date and time the task is expected to commence (advisory only)	UTC date time string in ISO8601 format, i.e. `yyyy-mm-ddThh:mm:ssZ`

Item	Description	Constraints
TargetFinish	The target date and time the task is expected to finish (advisory only)	UTC date time string in ISO8601 format, i.e. `yyyy-mm-ddThh:mm:ssZ`
TerminateExpiry	The date and time by which the user must complete or abandon the task (after which the task may be rescinded)	UTC date time string in ISO8601 format, i.e. `yyyy-mm-ddThh:mm:ssZ`
IconPngBase64	Base64 encoded png image data for the form's icon shown to user. If omitted, a standard icon will be used	20,000 chars max. Recommend keep as compact as possible.
TaskStateData	Optional arbitrary data the integrator may wish to include in the task, this is returned with the task when sent back. E.g. third-party reference IDs etc. Ignored by the system.	2,000 chars max.

8.3.7. Date and time conventions

To avoid ambiguity, all date and time properties are in UTC, except dates and times presented to users in form fields, which are always in local time, for the convenience of users.

To specify a local time in a field, use the ISO 8601 format with 'L' in place of 'Z' to indicate local time. i.e. `yyyy-mm-ddThh:mm:ssL`

The system will treat the local time verbatim without applying daylight saving or time zone changes when later accessed, allowing non-ambiguous use across time zones. The time value returned will always be the same as it was when recorded.

8.3.8. Field data types supported

The API currently supports the following datatype names for use in fields:

- Text: plain text
- Label/TextReadOnly: Like a text field but shows pre-populated text which can't be edited
- Password: plain text, obfuscated in user interface
- Time: Local time
- Date: Local date
- YesNo: Yes, No or empty values (ie Boolean plus null)
- ImageEmbedded: Embeds an image directly in the field as
- Hyperlink: User can view or record a hyperlink and view with an embedded browser
- Location: User can view or record a location (either by setting a pin on a map or selecting current location).

The following fields can be specified through the API but cannot pre-populated:

- Image: Allows user to capture an image which is stored separately from the form
- Fingerprint: User can capture a fingerprint (similar to Image but uses fingerprint scanner instead of camera)
- Audio: user can record audio (audio is stored separately to the form)
- Signature: User can collect a manuscript signature. Prompt defines the declaration.

8.3.9. Pre-populating fields

Non-media datatypes store their data as text in the text property of the field.

The formats to use are as follows:

- Text, TextReadOnly, Password: Plain text.

- Time: Use the following format: `yyyy-mm-ddThh:mm:ssL` the date component is ignored and can be set to 1st January 1900.
- Date: Use the following format: `yyyy-mm-ddThh:mm:ssL` the time component is ignored and can be set to 00:00:00..
- Yes/No: Store 1 for Yes/True, 0 for No/False, Empty for unset.
- Image Embedded: Store the image caption as text.
- Hyperlink: Store the hyperlink in the text property.
- Location: A predefined location can be stored as JSON. [TBA]

8.3.10. Return value JSON structure:

If the call was successful, the reference ID and internal ID of the created task are returned:

```
{
  "RefId": "string",
  "Id": "string"
}
```

- RefId: The unique reference ID for the task which can be assigned by the caller. This is the value specified in the call, or a string representation of a new GUID if it was not specified. This can be used to refer to the task in other API calls.

- Id: This is the internal unique decimal ID of the task generated by the system.

8.4. Active task status

This returns the current status of all active tasks (tasks which have been assigned but not yet completed)

8.4.1. Swagger documentation

`/Swagger/ui/index#!/ActiveTaskSummaries/ActiveTaskSummaries_Get`

8.4.2. Call details

`GET: api/syskey/strongsyskey/ActiveTaskSummaries`

8.4.3. Returned JSON


```
[
  {
    "User": "string",
    "Resource": "string",
    "RefId": "string",
    "Id": "string",
    "Name": "string",
    "Inits": "string",
    "Description": "string",
    "ProtectiveMarking": "string",
    "LatitudeWgs84": 0,
    "LongitudeWgs84": 0,
    "Address": "string",
    "Created": "2019-06-04T09:35:27.216Z",
    "AcceptExpiry": "2019-06-04T09:35:27.216Z",
    "TargetCommence": "2019-06-04T09:35:27.217Z",
    "TargetFinish": "2019-06-04T09:35:27.217Z",
    "TerminateExpiry": "2019-06-04T09:35:27.217Z",
    "TaskState": "string",
    "IsRetrieved": true,
    "RescindState": "string",
    "LastDeviceUpdate": "2019-06-04T09:35:27.217Z"
  }
]
```


- Resource: the name of the resource allocated to the task
- RefId: The unique reference ID assigned to the task
- ID: The internal decimal ID of the task (as a string)
- Name: The name of the type of task
- Inits: The initials of the type of task
- Description: Details of the task to brief the end user
- ProtectiveMarking: Protective marking description if required e.g. RESTRICTED
- LatitudeWgs84: The target latitude of the task, using WGS84 coordinate system
- LongitudeWgs84: The target longitude of the task, using WGS84 coordinate system
- Address: The target address of the task
- Created: UTC datetime when the task was created (UTC)
- AcceptExpiry: When the task needs to be accepted or rejected by (UTC)
- Target Commence: When the task is due to commence (UTC)
- TargetFinish: When the task is due to finish (UTC)
- TerminateExpiry: When the task must be completed or abandoned by (UTC)

- TaskState: The current status of the task(Unaccepted, Rejected, Accepted, Completed,Abandoned)
- IsRetrieved: If true, the mobile app has retrieved the task
- RescindState: (Empty, Requested or Rescinded)
- LastDeviceUpdate: When the device last updated the status (UTC)

8.5. Active task status for specified task

As per active task status, but for the specified RefID. Returns only the specified task.

8.5.1. Swagger documentation

`/Swagger/ui/index#!/ActiveTaskSummaries/ActiveTaskSummaries_Get_0`

8.5.2. Call details

`GET: api/syskey/strongsyskey/ActiveTaskSummaries/{RefId}`

8.6. Resources Status

Returns a summary of the current status of each resource/callsign.

8.6.1. Swagger documentation

`/Swagger/ui/index#!/ResourcesStatus/ResourcesStatus_Get`

8.6.2. Call details

`GET: /api/syskey/strongsyskey/ResourcesStatus + optional query string`

8.6.3. Query parameters (optional)

- availCode: Restrict to resources with the specified availability code*
- resourceName: Restrict to the specified resource name

8.6.4. Returned JSON

```
[
  {
    "Resource": "string",
    "Availability": "string",
    "AvailabilityDateTime": "2019-06-04T09:35:27.348Z",
    "LatitudeWgs84": 0,
    "LongitudeWgs84": 0,
    "LocationDateTime": "2019-06-04T09:35:27.348Z"
  }
]
```

- Resource: The name of the resource or callsign
- Availability: A code* indicating resource's last reported availability
- AvailabilityDateTime: Date/time when resource last reported availability (UTC)
- LatitudeWgs84: Resource's last reported latitude, WGS84 coordinate system
- LongitudeWgs84: Resource's last reported longitude, WGS84 coordinate system
- LocationDateTIme: Date/time when resource last reported location (UTC)

* - the availability codes are defined as required in the system configuration as set up by the customer

8.7. Users Status

Similar to resources status, returns a summary of the status of each user. It is recommended to use resource status unless there is an explicit need to handle user identities.

8.7.1. Swagger documentation

`/Swagger/ui/index#!/UsersStatus/UsersStatus_Get`

8.7.2. Call details

`GET: /api/syskey/strongsyskey/UsersStatus + optional query string`

8.7.3. Query parameters (optional)

- availCode: Restrict to users with the specified availability code*
- userName: Restrict to the specified user name

8.7.4. Returned JSON

This is similar to ResourcesStatus but includes the user name.

```
[
  {
    "User": "string",
    "Resource": "string",
    "Availability": "string",
    "AvailabilityDateTime": "2019-06-04T09:35:27.453Z",
    "LatitudeWgs84": 0,
    "LongitudeWgs84": 0,
    "LocationDateTime": "2019-06-04T09:35:27.453Z"
  }
]
```

- User: The user name of the user
- Resource: The name of the resource or callsign
- Availability: A code* indicating resource's last reported availability
- AvailabilityDateTime: Date/time when resource last reported availability (UTC)
- LatitudeWgs84: Resource's last reported latitude, WGS84 coordinate system
- LongitudeWgs84: Resource's last reported longitude, WGS84 coordinate system
- LocationDateTIme: Date/time when resource last reported location (UTC)

* - the availability codes are defined as required in the system configuration as set up by the customer

8.8. Manage resource names

Allows a master list of resource names to be synchronised with the system's list. Optionally, deletes can be enabled. This does not assign the resources to users. (See UsersResources).

8.8.1. Swagger documentation

`/Swagger/ui/index#!/Resources/Resources_Patch`

8.8.2. Call details

`PATCH: /api/{sysKey}/{strongSysKey}/Resources`

8.8.3. Query parameters (optional)

- enableDelete: If true, records in the master list which are not in the submitted list will be deleted. Set to false if the submitted list contains only new records to be added.

8.8.4. Submitted data

```
[
  {
    "ResourceName": "string"
  }
]
```

- ResourceName: the name of a resource (up to 75 chars)

8.8.5. Returned data

This confirms the operations performed.

```
[
  {
    "MergeAction": "string",
    "SourceResourceName": "string",
    "TargetResourceName": "string"
  }
]
```

- MergeAction: Insert, Update or Delete.
- Source ResourceName: If Inserting or updating, the source value used
- TargetResourceName: If deleting, the target value that was deleted

8.9. Manage Resource Capabilities

Allows a list of resource – capability pairs to be synchronised with the system's list. This associates resources with capabilities. If the capability refers to individual users, use 'Manage User Capabilities' instead.

8.9.1. Swagger documentation

`/Swagger/ui/index#!/ResourcesCapabilities/ResourcesCapabilities_Patch`

8.9.2. Call details

`PATCH: /api/{sysKey}/{strongSysKey}/ResourcesCapabilities`

8.9.3. Query parameters (optional)

- enableDelete: If true, capabilities in the master list which are not in the submitted list will be deleted (the parent resources are not affected). Set to false if the submitted list contains only new records to be added.

8.9.4. Submitted data

```
[
  {
    "ResourceName": "string",
    "ResourceCapability": "string"
  }
]
```

- ResourceName: The resource the capability applies to.
- ResourceCapability: Description of the capability (16 chars max)

8.9.5. Returned data

This confirms the operations performed.

```
[
  {
    "MergeAction": "string",
    "SourceResourceName": "string",
    "SourceResourceCapability": "string",
    "TargetResourceName": "string",
    "TargetResourceCapability": "string"
  }
]
```

- MergeAction: Insert, Update or Delete.
- Source ResourceName: If Inserting or updating, the source value used
- SourceResourceCapability: If inserting or updating, the source value used
- TargetResourceName: If deleting, the target value that was deleted
- TargetResourceCapability: If deleting, the target value that was deleted

8.10. Manage User Capabilities

Allows a list of user – capability pairs to be synchronised with the system's list. This associates users with capabilities. If the capability refers to resources rather than individual users, use 'Manage Resource Capabilities' instead.

8.10.1. Swagger documentation

`/Swagger/ui/index#!/UsersCapabilities/UsersCapabilities_Patch`

8.10.2. Call details

`PATCH: /api/{sysKey}/{strongSysKey}/UsersCapabilities`

8.10.3. Query parameters (optional)

- enableDelete: If true, records in the master list which are not in the submitted list will be deleted. Set to false if the intention is only to add new records.

8.10.4. Submitted data

```
[
  {
    "UserName": "string",
    "UserCapability": "string"
  }
]
```

- UserName: The user the capability applies to.
- UserCapability: Description of the capability (16 chars max)

8.10.5. Returned data

This confirms the operations performed.

```
[
  {
    "MergeAction": "string",
    "SourceUserName": "string",
    "SourceUserCapability": "string",
    "TargetUserName": "string",
    "TargetUserCapability": "string"
  }
]
```

- MergeAction: Insert, Update or Delete.
- SourceUserName: If Inserting or updating, the source value used
- SourceUserCapability: If inserting or updating, the source value used
- TargetUserName: If deleting, the target value that was deleted
- TargetUserCapability: If deleting, the target value that was deleted

8.11. Assign resources to users

Allows resources to be assigned to users. To avoid possible ambiguities, conflicts and duplications when processing tasks in a semi-offline system, only one user can be assigned to each resource.

If more than one user shares a resource name or callsign, then one user needs to be responsible for receiving tasks from the PoliceBox platform on behalf of that callsign.

If it is necessary to assign tasks to more than one user within the same callsign, assign the task explicitly to the relevant user instead of the callsign.

8.11.1. Swagger documentation

`Swagger/ui/index#!/UsersResources/UsersResources_Post`

8.11.2. Call details

`POST: /api/{sysKey}/{strongSysKey}/UsersResources`

8.11.3. Optional parameter

- ForceUnique: If true (the default value), the call will delete any user assignments to this resource name before applying this assignment. If false, multiple users can eb assigned to the same resource name.

8.11.4. Submitted data

```
{
  "UserName": "string",
  "ResourceName": "string"
}
```

- UserName: The user to assign the resource name to
- ResourceName: The resource name to assign

8.11.5. Response

OK (Code 200) if successful, otherwise an error code.

8.12. Get all users' resource assignments

Returns a list of current resource name assignments to users, as a list of user name / resource name pairs.

8.12.1. Swagger documentation

`/Swagger/ui/index#!/UsersResources/UsersResources_Get`

8.12.2. Call details

`GET: /api/{sysKey}/{strongSysKey}/UsersResources`

8.12.3. Response

```
[
  {
    "UserName": "string",
    "ResourceName": "string"
  }
]
```

- UserName: The login name of the user
- ResourceName: The corresponding resource name (empty string if not assigned)

8.13. Get specified user's resource assignment

Returns a single resource assignment for the specified user.

8.13.1. Swagger documentation

`/Swagger/ui/index#!/UsersResources/UsersResources_Get_0`

8.13.2. Call details

`GET: /api/{sysKey}/{strongSysKey}/UsersResources/{id}`

- Id: The specified user's login name

8.13.3. Response

`"string"`

- String contains the resource name allocated to the user (empty if not allocated)

9. Securing the API

The API employs multiple protection mechanisms which can be configured according to customer requirements. See the separate document “PoliceBox API Security and Data Integrity Protections”

[End of Document]