Qualcomm Technologies, Inc.

DIRBS Core API Specification

API-Specification-Document-1.0.0

Dec 10, 2018

Revision history

Revision	Date	Description
A	Dec 10, 2018	Initial release

Contents

Tables

1. Introduction

1.1. Purpose

The purpose of this document is to enlist all APIs of DIRBS Core in detail

1.2. Technical Stakeholders

This document is intended to be used by technical stakeholders of the DIRBS Core who will be responsible for planning, performing, or maintaining all API's such as the System Administrators, Analysts and Developers.

1.3. Definitions, Acronyms, and Abbreviations

Table 1-Definitions, Acronyms, and Abbreviations

Term	Definition
DIRBS	**Device Identification Registration and Blocking System**
API	Application Programming Interface
TAC	Type Allocation Code
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
MSISDN	Mobile Subscriber Integrated Services Digital Network Number

2. API v1.0

2.1. Version API

Version API shows the information DIRBS Core knows about the code, database schema and report schema version.

- **Implementation Notes**
  Information about the code and DB schema version used by Core.
- **API URL:** */api/v1/version*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**
  Following table shows the API responses in different scenarios.

Table 2-Version API Responses

Status Code	Response	Description
200	{ "code_db_schema_version": 0, "db_schema_version": 0, "report_schema_version": 0, "source_code_version": "string" }	This shows successful response from the API which shows the relevant details which are being fetched from the Core database and source code.
400	**Method Not Allowed** The method is not allowed for the requested URL.	This response is shown on any other http method than **GET**, which are not supported on this API.

2.2. TAC API

TAC API shows the information about the **TAC** in **GSMA database**.

- **Implementation Notes**

  Fetch TAC information from both the GSMA database.

- **API URL*:*** */api/v1/tac/<tac>*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**

  Following table shows the API responses in different scenarios.

Table 3-TAC API Responses

Status Code	Request Params	Request Method	API URL	Response	Description

200	TAC	GET	/api/v1/tac/
			<tac>
				{
				"gsma": {
				"allocation_date": "string",
				"bands": "string",
				"bluetooth": "string",
				"brand_name": "string",
				"country_code": "string",
				"device_type": "string",
				"fixed_code": "string",
				"internal_model_name": "string",
				"manufacturer": "string",
				"manufacturer_code": "string",
				"marketing_name": "string",
				"model_name": "string",
				"nfc": "string",
				"operating_system": "string",
				"radio_interface": "string",
				"wlan": "string"
				},
				"tac": "string"
				}
					This shows successful response from the API when a valid single TAC has been passed as parameter.

200	TAC	GET	/api/v1/tac/
			<tac>
				{
				"gsma": null,
				"tac": "string"
				}
					This response shows successful response from the API. It shows that the TAC information is not found in GSMA database.

400	TAC	GET	/api/v1/tac/ <tac>	**Bad Request** Bad TAC format	This response shows the TAC is not the the required format i.e. it is not numeric or don't have the required length (8 digits)
405	TAC	POST, PUT, DELETE, PATCH	/api/v1/tac/ <tac>	**Method Not Allowed** The method is not allowed for the requested URL.	This response shows that the referenced http method is not allowed.

2.3. MSISDN API

MSISDN API shows information core knows about the MSISDN. It fetches IMEI, IMSI, Manufacturer and Model information from GSMA database.

- **Implementation Notes**
  Information Core knows about the MSISDN. It returns IMEI, IMSI, GSMA Manufacturer, GSMA Model Name for the MSISDN specified.
- **API URL:** */api/v1/msisdn/<msisdn>*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**
  Following table shows the API responses in different scenarios.

Table 4-MSISDN API Responses

Status Code	Request Params	Request Method	API URL	Response	Description
200	MSISDN	GET	/api/v1/msisdn/<msisdn>	{ "gsma_manufacturer": "string", "gsma_model_name": "string", "imei_norm": "string", "imsi": "string" }	This shows successful response from the API.
400	MSISDN	GET	/api/v1/msisdn/<msisdn>	**Bad Request** Bad MSISDN format (can only contain digit characters)	This shows that the MSISDN is not in the required format i.e. length limit exceeds 15 digits or not valid digits.
405	MSISDN	POST, PUT, DELET, PATCH	/api/v1/msisdn/<msisdn>	**Method Not Allowed** The method is not allowed for the requested URL.	This response is shown on any other http method than **GET**, which are not supported on this API.

2.4. IMEI API

IMEI API shows information about IMEI such as its classification conditions, registration status, stolen status, real-time checks, subscribers list and pairs list. It support the following http methods:

- **Implementation Notes:**
  Information Core knows about the IMEI, as well as the results of all 'conditions' evaluated as part of DIRBS core. Calling systems should expose as little or as much of this information to the end user as is appropriate.
- **API URL:** */api/v2/imei/<imei>*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**

  Following table shows the API responses in different scenarios:

Table 5-IMEI API Responses

Status Code	Request Params	Request Method	API URL	Response	Description
200	IMEI	GET	/api/v1/ imei/ <imei>	{ "classification_state": { "blocking_conditions": { "compound_dimension": false, "simple_dimension": false }, "informative_conditions": {} }, "imei_norm": "string", "is_paired": false, "realtime_checks": { "ever_observed_on_network": false, "gsma_not_found": true, "in_registration_list": false, "invalid_imei": false } }	This shows a successful response from the API based on a valid IMEI. It shows the following information: **1. IMEI_norm** shows normalized form of the IMEI **2. classification_state** object shows configured classification state conditions as blocking conditions and informative conditions. Each object shows condition name and whether its met with the IMEI or not. **3. realtime_checks** shows different checked performed on the IMEI in real time it includes, if the IMEI is ever observed on network, if the IMEI is in invalid format, if the IMEI belongs to a exempted device type, and if the IMEI is paired.

200	IMEI	GET	api/v1/ imei/ <imei> ? include_seen_with	{ "classification_state": { "blocking_conditions": { "compound_dimension": false, "simple_dimension": false }, "informative_conditions": {} }, "imei_norm": "string", "is_paired": false, "realtime_checks": { "ever_observed_on_network": false, "gsma_not_found": true, "in_registration_list": false, "invalid_imei": false }, "seen_with": [] }	This shows a successful response from API for a random IMEI with its seen_with information
200	IMEI	GET	api/v1/ imei /<imei> ? include_paired_with	{ "classification_state": { "blocking_conditions": { "compound_dimension": false, "simple_dimension": false }, "informative_conditions": {} }, "imei_norm": "string", "is_paired": false, "paired_with": [], "realtime_checks": { "ever_observed_on_network": false, "gsma_not_found": true, "in_registration_list": false, "invalid_imei": false } }	This shows a successful response from API for a random IMEI with its paired_with information.
400	IMEI	GET	/api/v1 /imei/ <imei>	**Bad Request Bad** IMEI format (too long)	This shows an invalid IMEI format supplied to the API i.e. longer than 16 characters.
405	TAC	POST, PUT, DELETE, PATCH	/api/v1 /imei	**Method Not Allowed** The method is not allowed for the requested URL	This response show that the http method is not allowed on this endpoint.

2.5. Job Metadata API

Job Metadata API shows information about the jobs of the DIRBS Core, such jobs that are currently running, successful jobs, failed jobs. It also shows detailed information about each job. This API supports following http methods:

- **Implementation Notes**
  Information Core knows about the DIRBS jobs run on the system. It is intended to be used by operational staff to generate data for the admin panel.
- **API URL:** */api/v1/job_metadata*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **Supported Request Parameters:**
  Job metadata API supports the following request parameters:
  - **show_details** (*boolean*): Whether or not to include 'extra_metadata' field in the results
  - **subcommand** (Array[string]): Filter results to include only jobs belonging to specified subcommand(s)
  - **status** (Array[string]): Filter results to only include jobs having the specified status
  - **command** (Array[string]): Filter results to include only jobs belonging to specified command(s)
  - **run_id** (Array[string]): Filter results to only include job with the specified run_id(s)
  - **max_results** (Integer): Number of jobs to return sorted by run_id in descending order
- **API Responses:**
  Following table shows the API responses in different scenarios.

Table 6- Job Metadata API Responses

Status Code	Request Params	Request Method	API URL	Response	Description
200	Request Params	GET	api/v1/job_ metadata	{ "command": "string", "command_line": "string", "db_user": "string", "end_time": "DateTime", "exception_info": "string", "extra_metadata": {}, "run_id": 0, "start_time": "DateTime", "status": "string", "subcommand": "string" }	This shows a successful response from the API.
400	Request Params	GET	api/v1/job_ metadata	**Bad Request**	This shows a 400, Bad Request response from the API.

405	Request Params	POST, PUT, DELET, PATCH	api/v1/job_ metadata	**Method Not Allowed** The method is not allowed for the requested URL.	This shows a 405, Method Not Allowed response from the API. It will be generated in-case of any other http method than GET being used.

2.6. Catalog API

Catalog API shows information Core knows about the cataloged data files. It returns a list of files along with their properties and state of validation checks run by Core. This API supports following http methods:

- **Implementation Notes**
  Information Core knows about the cataloged data files. It returns a list of files along with their properties and state of validation checks run by Core
- **API URL:** */api/v1/catalog*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **Supported Request Parameters:**
  Catalog API supports the following request parameters:
  - **file_type** Filter results to include only the specified file type
  - **is_valid_zip** Filter results to include only valid ZIP files
  - **modified_since** Filter results to only include files that were modified since the specified time
  - **cataloged_since** Filter results to include only files that were cataloged since the specified time
  - **max_results** Number of entries to return sorted by last_seen timestamp in descending order
- **API Responses:**
  Following table shows the API responses in different scenarios.

Table 7- Catalog API Responses

Status Code	Request Params	Request Method	API URL	Response	Description

200	Request Params	GET	api/v1/ catalog	{
				"catalog": [{
				"compressed_size_bytes": 0,
				"extra_attributes": {},
				"file_id": 0,
				"file_type": "string",
				"filename": "string",
				"first_seen": "DateTime",
				"import_status": {},
				"is_valid_format": true,
				"is_valid_zip": true,
				"last_seen": "DateTime",
					This shows a successful response from the API.

				"md5": "string", "modified_time": "DateTime", "num_records": 0, "uncompressed_size_bytes": 0 }] }	
400	Request Params	GET	api/v1/ catalog	**Bad Request**	This shows a 400, Bad Request response from the API. It will be generated in-case of any input/request parameters be not in the expected/correct format.
405	Request Params	POST, PUT, DELET, PATCH	api/v1/ catalog	**Method Not Allowed** The method is not allowed for the requested URL.	This shows a 405, Method Not Allowed response from the API. It will be generated in-case of any other http method than GET being used.

3. API v2.0

3.1. Version API

Version API shows the information DIRBS Core knows about the code, database schema version and reports schema version.

- **Implementation Notes**
  Information about the code, DB schema and reports schema version used by Core.
- **API URL:** */api/v2/version*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**
  Following table shows the API responses in different scenarios.

Table 8-Version API Responses 2.0

Status Code	Response	Description
200	{ "code_db_schema_version": 0, "db_schema_version": 0, "report_schema_version": 0, "source_code_version": "string" }	This shows successful response from the API which shows the relevant details which are being fetched from the Core database and source code.
405	**Method Not Allowed** The method is not allowed for the requested URL.	This response is shown on any other http method than **GET**, which are not supported on this API.

3.2.TAC API

TAC api shows the information about the **TAC** in **GSMA database.**

- **Implementation Notes**
  Fetch TAC information from GSMA database.
- **API URL**
  - */api/v2/tac/<tac>*
  - */api/v2/tac*
- **Supported Methods:** *GET, POST*
- **Supported Content Type:** *application/json*
- **API Responses**
  Following table shows the API responses in different scenarios.

Table 9-TAC API Responses 2.0

Status Code	**Request Params**	**Request Method**	**API URL**	**Response**	**Description**
200	TAC	GET	/api/v2/tac/<tac>	{ "gsma": { "allocation_date": "string", "bands": "string", "bluetooth": "string", "brand_name": "string", "device_type": "string", "internal_model_name": "string", "manufacturer": "string", "marketing_name": "string", "model_name": "string", "nfc": "string", "radio_interface": "string", "wlan": "string" }, "tac": "string" }	This shows a successful response from the API.
200	TAC	GET	/api/v2/tac/<tac>	{ "gsma": null, "tac": "string" }	This response shows successful response from the API. It shows that the TAC information is not found in GSMA database.
400	TAC	GET	/api/v2/tac/<tac>	**Bad Request** Bad TAC format	This response shows the TAC is not the required format i.e it is not numeric or don't have the required length (8 digits)
405	TAC	POST, PUT, DELET, PATCH	/api/v2/tac/<tac>	**Method Not Allowed** The method is not allowed for the requested URL.	This shows a 405, Method Not Allowed response from the API. It will be generated in-case of any other http method than GET being used.

200	{
	"tacs": [
	"tac 1",
	"tac 2",
	"tac 3",
	.....
	]
	}
		POST	/api/v2/tac	{
				"results": [
				{
				"gsma": {
				"allocation_date": "string",
				"bands": "string",
				"bluetooth": "string",
				"brand_name": "string",
				"device_type": "string",
				"internal_model_name": "string",
				"manufacturer": "string",
				"marketing_name": "string",
				"model_name": "string",
				"nfc": "string",
				"radio_interface": "string",
				"wlan": "string"
				},
				"tac": "string"
				}
				]
				}
					This shows successful response from the API. The **results** object contain the data about the each TAC from GSMA database.

400	{
	"tacs": [
	"tac 1",
	"tac 2",
	"tac 3",
	.....
	]
	}
		POST	/api/v2/tac	**Bad Request**
				Bad TAC format
					This response shows the TACs are not in the required format i.e. it is not numeric, don't have the required length (8 digits) or number of tacs are not in the required limit (maximum 1000).

405	{ "tacs": ["tac 1", "tac 2", "tac 3", ...]}	POST, PUT, DELET, PATCH	/api/v2/tac	**Method Not Allowed** The method is not allowed for the requested URL.	This response shows that the referenced http method is not allowed on the API.

3.3. MSISDN API

MSISDN API shows information core knows about the MSISDN. It fetches IMEI, IMSI, Last Seen (Brand, Manufacturer and Model) information from GSMA database and (brand, make and model) from Device Registration System.

- **Implementation Notes**
  Information Core knows about the MSISDN. It returns a list of IMEI, IMSI, GSMA Manufacturer, GSMA Model Name and information from Device Registration System for the MSISDN specified.
- **API URL:** */api/v2/msisdn/<msisdn>*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **API Responses**
  Following table shows the API responses in different scenarios.

Table 10-MSISDN API Responses 2.0

Status Code	Request Params	Request Method	API URL	Response	Description
200	MSISDN	GET	/api/v2/msisdn/<msisdn>	{ "results":[{ "imei_norm":"string", "imsi":"string", "gsma":{ "manufacturer":"string" "model_name":"string", "brand_name":"string" }, "registration":{ "manufacturer":"string", "model_name":"string", "brand_name":"string" }, "last_seen":"Date" }] }	This shows successful response from the API. It shows that the MSISDN information is found in the GSMA database and device registration system. Multiple data objects can exists in the **results** object in case of multiple occurrences of msisdn in core.

200	MSISDN	GET	/api/v2/msisdn/
			<msisdn>
				{
				"results":[
				{
				"imei_norm":"string",
				"imsi":"string",
				"gsma":
				{
				"manufacturer":"string",
				"model_name":"string",
				"brand_name":"string"
				},
				"registration":null,
				"last_seen":"Date"
				}
				]
				}
					This shows successful response from the API. It shows that the MSISDN information is found in the GSMA database and but not found in the device registration system. Multiple data objects can have similar behaviors and behaviors as in previous scenario shown in above row.

200	MSISDN	GET	/api/v2/msisdn/
			<msisdn>
				{
				"results":[]
				}
					This shows successful response from the API. It shows that a valid MSISDN information could not be found in the core.

400	MSISDN	GET	/api/v2/msisdn/ <msisdn>	**Bad Request** Bad MSISDN format (can only contain digit characters)	This shows that the MSISDN is not in the required format i.e. length limit exceeds 15 digits or not valid digits.
405	MSISDN	POST, PUT, DELETE, PATCH	/api/v2/msisdn/ <msisdn>	**Method Not Allowed** The method is not allowed for the requested URL.	This response is shown on any other http method than **GET**, which are not supported on this API.

3.4. IMEI API

IMEI API shows information about IMEI such as its classification conditions, registration status, stolen status, real-time checks, subscribers list and pairs list.

- **Implementation Notes:**
  - **IMEI:** Information Core knows about the IMEI, as well as the results of all 'conditions' evaluated as part of DIRBS core. Calling systems should expose as little or as much of this information to the end user as is appropriate.
  - **IMEI Info:** Information Core knows about the details related to an IMEI such as brand name, device type, make, model, model number, radio interfaces and current status of the IMEI.
  - **IMEI-Subscribers:** Information Core knows about the IMSI-MSISDN pairs the IMEI has been seen with on the network. The results are returned in paginated format.
  - **IMEI-Pairings:** Information Core knows about the IMSIs paired with the IMEI in the device pairing system. The results are returned in paginated format.
  - **IMEI-Batch:** Information Core knows about each IMEI (max:1000) in the batch request, as well as the results of all 'conditions' evaluated as part of DIRBS core.
- **API URL:**
  - */api/v2/imei/<imei>*
  - */api/v2/imei/<imei>/info*
  - */api/v2/imei/<imei>/subscribers*
  - */api/v2/imei/<imei>/pairings*
  - */api/v2/imei-batch*
- **Supported Methods:** *GET, POST*
- **Supported Content Type:** *application/json*

- **API Responses:**
  Following table shows the API responses in different scenarios.

Table 11- IMEI API Responses 2.0

Status Code	Request Params	Request Method	API URL	Response	Description
200	IMEI	GET	api/v2/imei/ <imei>	{ "imei_norm":"string", "block_date": "Date", "classification_state":{ "blocking_conditions":[{ "condition_name":"string", "condition_met":true }], "informative_conditions":[{ "condition_name":"string", "condition_met":true }]}, "realtime_checks":{ "ever_observed_on_network":true, "invalid_imei":true, "is_exempted_device": true, "is_paired":true }, "registration_status":{ "status":"string", "provisional_only":true } "stolen_status":{ "status":"string", "provisional_only":true } }	This shows a successful response from the API based on a valid IMEI. It shows the following information: 1. **IMEI_norm** shows normalized form of the IMEI 2. **classification_state** object shows configured classification state conditions as blocking conditions and informative conditions. Each object shows condition name and whether it's met with the IMEI or not. 3. **realtime_checks** shows different checked performed on the IMEI in real time it includes, if the IMEI is ever observed on network, if the IMEI is in invalid format, if the IMEI belongs to an exempted device type, and if the IMEI is paired. 4. **registration_status** shows information about IMEI from device registration system, it includes status of the IMEI and whether the IMEI is provisionally registered or not. 5. **stolen_status** shows information about IMEI from stolen list management system, it includes status of the IMEI and whether the IMEI is in the list temporarily or not.

200	IMEI	GET	api/v2/imei/ <imei>	{ "imei_norm":"string", "classification_state":{ "blocking_conditions":[], "informative_conditions":[] }, "realtime_checks":{ "ever_observed_on_network":false, "invalid_imei":false, "is_exempted_device": false, "is_paired":false }, "registration_status": { "status":null, "provisional_only":null } , "stolen_status": { "status":null, "provisional_only":null } }	This shows a successful response from API for a random IMEI, which is not observed on network, is in valid format, is not in exempted device list, is not paired, have no information in device registration system and stolen list management system. It also shows that there are no blocking or informative conditions configured on the DIRBS.
200	IMEI	GET	api/v2/imei/ <imei>	{ "imei_norm":"string", "classification_state": { "blocking_conditions":[{ "condition_name": "name", "condition_met": false }, { "condition_name": "name", "condition_met": true }], "informative_conditions":[{ "condition_name":"name", "condition_met":false }] }, "realtime_checks":{ "ever_observed_on_network":true, "invalid_imei":false, "is_exempted_device": false, "is_paired":true }, "registration_status": { "status":"whitelist", "provisional_only":false }, "stolen_status": { "status":null, "provisional_only":null } }	This shows a successful response from API provided a valid IMEI. It shows two blocking conditions and one informative condition, the IMEI met on of the blocking condition and informative condition is not met. It also shows that the IMEI is observed on the network before, is paired and not in the exempted device list. It also shows the IMEI registration status from device registration system which is whitelist and it is not provisionally registered. **Note:** All the parts of the IMEI response will behave similarly.

200	IMEI	GET	api/v2/imei/ <imei>	{ "classification_state"{ "blocking_conditions": [], "informative_conditions": [] }, "imei_norm": "string", "realtime_checks"{ "ever_observed_on_network": false, "invalid_imei": true, "is_exempted_device": false, "is_paired": false }, "registration_status":{ "provisional_only": null, "status": null } , "stolen_status": { "provisional_only": null, "status": null } }	This shows a successful response from API in-case of short length IMEI i.e. 123456. It shows **invalid_imei** check evaluated to **true**.
200	IMEI	GET	api/v2/imei/ <imei>	{ "block_date": "Date", "classification_state"{ "blocking_conditions": [{}], "informative_conditions": [{}] }, "imei_norm": "string", "realtime_checks"{ "ever_observed_on_network": true, "invalid_imei": true, "is_exempted_device": true, "is_paired": true }, "registration_status":{ "provisional_only": true, "status": "string" }, "stolen_status": { "provisional_only": true, "status": "string" } }	The response shows an IMEI details along with a **block_date** parameter set. It will be either on notification lists or black list.
200	IMEI	GET	api/v2/imei/ <imei>	{ "classification_state"{ "blocking_conditions": [{}], "informative_conditions": [{}] }, "imei_norm": "string", "realtime_checks":{ "ever_observed_on_network": true, "invalid_imei": true, "is_exempted_device": true, "is_paired": true }, "registration_status"{ "provisional_only": true, "status": "string" }, "stolen_status": { "provisional_only": true, "status": "string" } }	The response show the IMEI details but no **block_date** parameter can be observed this means that the IMEI is either on notifications lists nor on black list.

400	IMEI	GET	api/v2/imei/ <imei>	**Bad Request** Bad IMEI format (too long)	This shows an invalid IMEI format supplied to the api i.e longer than 16 characters.

200	IMEI	GET	api/v2/imei/
			<imei>/info
				{
				"brand_name": "string",
				"device_type": "string",
				"imei_norm": "string",
				"make": "string",
				"model": "string",
				"model_number": "string",
				"radio_interface": "string",
				"status": "string"
				}
					This shows successful response from API. It includes Normalized form of the IMEI, its brand, type, make, model, model number, radio interface and current status of the IMEI

200	IMEI	GET	api/v2/imei/ <imei>/info	{ }	This shows a response where no data is found for that IMEI.
400	IMEI	GET	api/v2/ <imei>/info	Bad Request	This shows an invalid IMEI format supplied to the API i.e. longer than 16 characters.

200	IMEI	GET	api/v2/imei/
			<imei>/
			subscribers
				{
				"_keys":{
				"previous_key":"string",
				"next_key":"string",
				"result_size":0
				},
				"imei_norm":"string",
				"subscribers":[
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				}
				]
				}
					This shows successful response from API, it includes a paginated result of IMSI-MSISDN pairs seen with the IMEI.
					
					The **_keys** objects indicates the parameters of the paginated result such as **previous_key** indicates the key to the previous page result, **next_key** indicates the key to the next result in the paginated data, **IMEI_norm** indicates normalized form of the IMEI, **subscribers** object will display list of imsi-MSISDN pairs with last_seen date.

200	IMEI, offset=1, limit=3	GET	api/v2/imei/<imei>/subscribers	{
				"_keys":{
				"previous_key":"string",
				"next_key":"string",
				"result_size":10
				},
				"imei":"string",
				"subscribers":[
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				}
				]
				}
					This response shows that the given IMEI has 10 matched subscribers pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of three results per page.

200	IMEI, offset=1, limit=3, order=Ascending	GET	api/v2/imei/<imei>/subscribers	{
				"_keys":{
				"previous_key":"string",
				"next_key":"string",
				"result_size":10
				},
				"imei":"string",
				"subscribers":[
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"msisdn":"string",
				"last_seen":"Date"
				}
				]
				}
					This response shows that the given IMEI has 10 matched subscribers pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of three results per page sorted in **ascending** order based on **last_seen**.

200	IMEI, offset=1, limit=3, order=Descending	GET	api/v2/imei/<imei>/subscribers	{ "_keys":{ "current_key":"string", "next_key":"string", "result_size":10 }, "imei":"string", "subscribers":[{ "imsi":"string", "msisdn":"string", "last_seen":"Date" }, { "imsi":"string", "msisdn":"string", "last_seen":"Date" }, { "imsi":"string", "msisdn":"string", "last_seen":"Date" }] }	This response shows that the given IMEI has 10 matched subscribers pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of three results per page sorted in **descending** order based on **last_seen**.
400	IMEI	GET	api/v2/imei/<imei>/subscribers	**Bad Request** Bad IMEI format (too long)	This shows an invalid IMEI format supplied to the API i.e. longer than 16 characters.
200	IMEI	GET	api/v2/imei/<imei>/pairings	{ "_keys":{ "current_key":"string", "next_key":"string", "result_size":0 }, "imei_norm":"string", "pairs":[{ "imsi":"string", "last_seen":"Date" }] }	This shows successful response from API, it includes a paginated result of IMSI-MSISDN pairs seen with the IMEI. The **_keys** objects indicates the parameters of the paginated result such as **previous_key** indicates the key to the previous page result, **next_key** indicates the key to the next result in the paginated data, **IMEI_norm** indicates normalized form of the IMEI, **pairs** object will display list of imsi pairs with last_seen date.

200	IMEI, offset=1, limit=2	GET	api/v2/imei/<imei>/pairings	{
				"_keys":{
				"current_key":"string",
				"next_key":"string",
				"result_size":10},
				"imei":"string",
				"pairs":[
				{
				"imsi":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"last_seen":"Date"
				}
				]
				}
					This response shows that the given IMEI has 10 matched pairing pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of two results per page.

200	IMEI, offset=1, limit=2, order=Ascending	GET	api/v2/imei/<imei>/pairings	{
				"_keys":{
				"current_key":"string",
				"next_key":"string",
				"result_size": 10
				},
				"imei":"string",
				"pairs":[
				{
				"imsi":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"last_seen":"Date"
				}
				]
				}
					This response shows that the given IMEI has 10 matched subscribers pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of two results per page sorted in **ascending** order based on **last_seen**.

200	IMEI, offset=1, limit=2, order=Descending	GET	api/v2/imei/<imei>/pairings	{
				"_keys":{
				"current_key":"string",
				"next_key":"string",
				"result_size": 10
				},
				"imei":"string",
				"pairs":[
				{
				"imsi":"string",
				"last_seen":"Date"
				},
				{
				"imsi":"string",
				"last_seen":"Date"
				}
				]
				}
					This response shows that the given IMEI has 10 matched subscribers pairs and as per input parameters set by us as offset and limit, it will start displaying results from first pair and show total number of two results per page sorted in **descending** order based on **last_seen**.

400	IMEI	GET	api/v2/imei/<imei>/pairings	**Bad Request** Bad IMEI format (too long)	This shows an invalid IMEI format supplied to the API i.e. longer than 16 characters.

200	{
	"imeis":[
	"string"
	]
	}
		POST	api/v2/imei-batch	{
				"results":[
				{
				"imei_norm":"string",
				"classification_state"{
				"blocking_conditions":[
				{
				"condition_name":"string",
				"condition_met":true
				}],
				"informative_conditions":[
				{
				"condition_name":"string",
				"condition_met":true
				}]
				},
				"realtime_checks":{
				"ever_observed_on_network":true,
				"invalid_imei":true,
				"is_paired":true },
				"registration_status":{
				"status":"string",
				"provisional_only":true }
				"stolen_status":{
				"status":"string",
				"provisional_only":true
				}
				}]
				}
					This shows a successful response from **IMEI-batch** , it accepts multiple IMEIs (max 1000) in the form for string list and respond back.
					
					The result object will contain information for each IMEI. IMEI information is same as in IMEI API.

200	{ "imeis":["imei1", "imei2”] }	POST	api/v2/imei-batch	{ "results": [{ "block_date": "Date", "classification_state"{ "blocking_conditions": [{}], "informative_conditions": [{}] }, "imei_norm": "string", "realtime_checks":{ "ever_observed_on_network": true, "invalid_imei": true, "is_exempted_device": true, "is_paired": true }, "registration_status":{ "provisional_only": true, "status": "string" }, "stolen_status": { "provisional_only": true, "status": "string" }}, { "classification_state":{ "blocking_conditions": [{}], "informative_conditions": [{}] }, "imei_norm": "string", "realtime_checks": { "ever_observed_on_network": true, "invalid_imei": true, "is_exempted_device": true, "is_paired": true }, "registration_status": { "provisional_only": true, "status": "string" }, "stolen_status": { "provisional_only": true, "status": "string" }}] }	The response shows details of two IMEIs, one of them have a **block_date** parameter set which means it is either on notification list or black list and the other one don't have a **block_date** parameter which shows it is not found on both of lists.
400	{"imeis": ["string"]}	POST	api/v2/imei-batch	**Bad Request** Bad IMEI format	This will show up in case to wrong IMEI format such as longer than 16 characters, empty IMEI etc.
400	{"imeis": ["string"]}	POST	api/v2/imei-batch	**Bad Request** Bad IMEI format (max allowed IMEIs are 1000)	This will show up in-case if total number of IMEIs are more than 1000.

3.5. Job Metadata API

Job Metadata API shows information about the jobs of the DIRBS Core, such jobs that are currently running, successful jobs, failed jobs. It also shows detailed information about each job. This API supports following http methods:

- **Implementation Notes**
  Information Core knows about the DIRBS jobs run on the system. It is intended to be used by operational staff to generate data for the admin panel.

- **API URL:**
  - */api/v2/job_metadata*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **Supported Request Parameters:**
  Job metadata API supports the following request parameters:
  - **show_details** (*boolean*): Whether or not to include 'extra_metadata' field in the results
  - **subcommand** (Array[string]): Filter results to include only jobs belonging to specified subcommand(s)
  - **status** (Array[string]): Filter results to only include jobs having the specified status
  - **command** (Array[string]): Filter results to include only jobs belonging to specified command(s)
  - **run_id** (Array[string]): Filter results to only include job with the specified run_id(s)
  - **offset** (integer): Offset the results on the current page by the specified run_id. It should be the value of run_id for the last result on the previous page
  - **limit** (integer): Number of results to return on the current page
  - **order** (string): The sort order for the results using run_id as the key

- **API Responses:**
  Following table shows the API responses in different scenarios.

Table 12-Job Metadata API Responses 2.0

Status Code	**Request Params**	**Request Method**	**API URL**	**Response**	**Description**
200	Request Params	GET	api/v2/ job_metadat a	`{ "_keys":{ "current_key":"string", "next_key":"string", "result_size":0 }, "jobs":[ { "command":"string", "command_line":"string", "db_user":"string", "end_time":"Date", "exception_info":"string", "extra_metadata":{}, "run_id":0, "start_time":"Date", "status":"string", "subcommand":"string" } ] }`	This shows a successful response from the API. The **_keys** object indicate the keys related to paginated result and the **jobs** object indicate the actual API response, it can contain multiple nested objects.
200	Request Params	GET	api/v2/ job_metadat a	`{ "_keys":{ "current_key":"string", "next_key":"string", "result_size":0 }, "jobs":[] }`	This shows a successful response from API but no jobs data has been found.

200	Request Params	GET	api/v2/job_ metadata	{ "_keys":{ "current_key":"string", "next_key":"string", "result_size":2 }, "jobs":[{ "command":"string", "command_line":"string", "db_user":"string", "end_time":"DateTime", "exception_info":"string", "extra_metadata":{}, "run_id":0, "start_time":"DateTime", "status":"string", "subcommand":"string" }, { "command":"string", "command_line":"string", "db_user":"string", "end_time":"DateTime", "exception_info":"string", "extra_metadata":{}, "run_id":0, "start_time":"DateTime", "status":"string", "subcommand":"string" }] }	This shows a successful response from API with two jobs result.
400	Request Params	GET	api/v2/job_ metadata	**Bad Request**	This shows a 400, Bad Request response from the API. It will be generated in-case of any input/request parameters be not in the expected/correct format.
405	Request Params	POST, PUT, DELETE, PATCH	api/v2/job_ metadata	**Method Not Allowed** The method is not allowed for the requested URL.	This shows a 405, Method Not Allowed response from the API. It will be generated in-case of any other http method than GET being used.

3.6.Catalog API

Catalog API shows information Core knows about the cataloged data files. It returns a list of files along with their properties and state of validation checks run by Core.

- **Implementation Notes**
  Information Core knows about the cataloged data files. It returns a list of files along with their properties and state of validation checks run by Core
- **API URL:** */api/v2/catalog*
- **Supported Methods:** *GET*
- **Supported Content Type:** *application/json*
- **Supported Request Parameters:**
  Catalog API supports the following request parameters:
  - **file_type** Filter results to include only the specified file type
  - **is_valid_zip** Filter results to include only valid ZIP files
  - **modified_since** Filter results to only include files that were modified since the specified time
  - **cataloged_since** Filter results to include only files that were cataloged since the specified time
  - **offset** Offset the results on the current page
  - **limit** Number of results to return on the current page
  - **order** The sort order (Ascending/Descending) for the results using file_id as the key

- **API Responses:**
  Following table shows the API responses in different scenarios:

Table 13-Catalog API Responses 2.0

Status Code	Request Params	Request Method	API URL	Response	Description
200	Request Params	GET	api/v2/catalog	{ "_keys": { "current_key": "string", "next_key": "string", "result_size": 0 }, "files": [{ "compressed_size_bytes": 0, "extra_attributes": {}, "file_id": 0, "file_type": "string", "filename": "string", "first_seen": "DateTime", "import_status": {}, "is_valid_format": true, "is_valid_zip": true, "last_seen": "DateTime", "md5": "string", "modified_time": "DateTime", "num_records": 0, "uncompressed_size_bytes": 0 }] }	This shows a successful response from the API. The **_keys** object indicate the keys related to paginated result and the **files** object indicate the actual API response, it can contain multiple nested objects.
200	Request Params	GET	api/v2/catalog	{ "_keys": { "current_key": "string", "next_key": "string", "result_size": 0 }, "files": [] }	This shows a successful response from API but no files data has been found.

200	Request Params	GET	api/v2/ catalog	{ "_keys": { "current_key": "string", "next_key": "string", "result_size": 0 }, "files": [{ "compressed_size_bytes": 0, "extra_attributes": {}, "file_id": 0, "file_type": "string", "filename": "string", "first_seen": "DateTime", "import_status": {}, "is_valid_format": true, "is_valid_zip": true, "last_seen": "DateTime", "md5": "string", "modified_time": "DateTime", "num_records": 0, "uncompressed_size_bytes": 0 }, { "compressed_size_bytes": 0, "extra_attributes": {}, "file_id": 0, "file_type": "string", "filename": "string", "first_seen": "DateTime", "import_status": {}, "is_valid_format": true, "is_valid_zip": true, "last_seen": "DateTime", "md5": "string", "modified_time": "DateTime", "num_records": 0, "uncompressed_size_bytes": 0 }] }	This shows a successful response from API with two files result.
400	Request Params	GET	api/v2/ catalog	**Bad Request**	This shows a 400, Bad Request response from the API. It will be generated in-case of any input/request parameters be not in the expected/correct format.
405	Request Params	POST, PUT, DELETE, PATCH	api/v2/ catalog	**Method Not Allowed** The method is not allowed for the requested URL.	This shows a 405, Method Not Allowed response from the API. It will be generated in-case of any other http method than GET being used.