ComPair Data

API Service

(On-demand schedule data)

Message Type:	**API Service Documentation**
Version:	**1.0.0c**
Document Issue Date:	**10 August 2018**
Document Version:	**9th Revision**
Document Issuer:	**ComPair Data, Inc.**

ComPair Data

Table of Contents

ComPair Data

Introduction

ComPair Data API is an on-demand, RESTful API service for querying information from the ComPair Data system. Data sets such as Proforma Schedules, Point to Point Schedules, Voyage Schedules, Vessels, Carriers, Capacity and Transit Reports are made available via a simple API request.

All API requests submitted to ComPair Data's API Service require a valid, private API key, which is issued to each customer. This API key allows customers to make authorized requests to the ComPair Data API Service. It is recommended to keep this key private to avoid unwanted usage charges or reach API Service limits without warning.

For more information on API keys, see ***Appendix G***.

> **Note:** During Beta release of ComPair Data's API Service, not all datasets may be available and dataset names, formats, and structure may change in production or later releases.

Basic API Request:

The following is a basic example of the syntax required to make an API request to ComPair Data's API Service.

https://api.bwrdata.com/get/json?key=<CUSTOMER_API_KEY>&dataset=services&service_id=4

In this example, a request is made to query services within the `services` dataset, where the records match a `service_id` value of 4. The text **<CUSTOMER_API_KEY>** represents the place where a valid API Key would be used. Refer to **Figure 1** for an explanation on each part of the example request.

Figure 1

https://api.bwrdata.com	The address of ComPair Data's API Service.
/get/	The type of request made to ComPair Data's API Service.
/json	The format of the data returned from ComPair Data's API Service. Alternatively, this could also be "/csv".
?key=<CUSTOMER_API_KEY>&dataset=services&service_id=4	Contains any additional parameters, including the search criteria passed along with the

	request to ComPair Data's API Service. The parameter "key" is required with each request. The parameter "dataset" identifies the data to which the search criteria (i.e.: "service_id") applies and is also required.

For all basic API requests to ComPair Data's API Service, parameters in ***Figure 2*** will always be required. Failure to include the following parameters will result in an error code 400 and a message indicating that the request is invalid. (For a complete list of available query parameters, see ***Appendix C***)

Figure 2

key	A unique customer authentication code like the following: X00073BA-0000-FE43-BE00-84000X5CD0XD
dataset	The name of the dataset within the ComPair Data system. All additional parameters are translated as search criteria for fields available to this dataset. Examples of datasets include "services", "voyageschedules", "capacity", "p2pschedules", etc.

Based on the dataset passed in the request, the search criteria can include any field available within the dataset. For example, the dataset "services" has fields which include "service_id", "service_name", "frequency", "vesseldeploymentarea", etc. Any of these fields can be passed within the request made to ComPair Data's API Service to filter for specific data.

Note [1]: The **amendment_code** parameter is available only for certain datasets such as voyage and point to point schedules. This helps filter requests to only certain amendment codes such as "A" (ADD), "U" (UPDATE), "D" (DELETE). Please refer to ***Appendix B*** for a definition of each type of amendment code.

Note [2]: It is recommended that parameter values passed in the API request are URL encoded if they contain any unsafe characters. The ComPair Data system will do its best to match based on the criteria provided, however some characters may be unsafe to pass within the request such as forward slashes (/) and may result in a mismatch or a failed request if not URL encoded, so URL encoding is advised. Please refer to ***Appendix E*** for a list of unsafe characters.

Note [3]: If no criteria are passed in the request, all available data within the dataset, up to the maximum allowed, will be sent back in the response from the ComPair Data API Service to the client system making the request. It is recommended to always pass criteria parameters, especially for the larger data sets, to ensure response data can be successfully downloaded. If the response data returned from ComPair Data's API Service is too large, the system could potentially timeout and the download will fail. As a rule of thumb, any response data larger than approximately 524,288KB (a little over half a gigabyte) could potentially fail.

Response data formatting

The following formats are supported for API requests made to ComPair Data's API Service:

<table>
<tr><td rowspan="2">/json</td><td rowspan="2">Data returned from a request made to ComPair Data's API Service will be in the form of a JSON string which translates to an array containing only a single item. The item is an object containing two parameters HEADER and DATA, with DATA being an array. Each item in the DATA array is an object representing a single record within the requested dataset. Response data in JSON format will always be in the form of an array unless otherwise stated in the dataset documentation or if the request results in an error.</td><td>JSON Response Example</td></tr>
<tr><td><pre>[{
	“HEADER”: {
		“RECORDS_RETURNED”: 1
	},
	“DATA”: [{
		"SERVICE_ID": 4,
		"ALLIANCE_ID": null,
		"SERVICE_NAME": "ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service",
		"SERVICE_LONG_NME": "ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service (Transatlantic)",
		"FREQUENCY": "7 days",
		"START_DOW": 4,
		"START_DT": "02/07/2018 00:00",
		"FREQUENCY_IN_DAYS": 7,
		"DIRECTION": "East-West",
		"VESSELDEPLOYMENTAREA": "ECNA-EUR-ECNA"
	}]
}]</pre></td></tr>
<tr><td>/csv</td><td colspan="2">Data returned from a request made to ComPair Data's API Service will be in plain text with a comma <code>,</code> (ASCII character code 44) as the delimiter and double-quotes <code>"</code> (ASCII character code 34) as a text-qualifier. By default, the text-qualifier applies to all fields. Each record will be separated by escape sequence <code>\n</code> (ASCII control character code 10) to represent end of line (EOL).</td></tr>
</table>

Note: The two Media Types (formerly known as MIME Types) for JSON and CSV are “application/json” and “text/plain” respectively. These Media Types will tell your application how to interpret the data it

receives. Please refer to the Media Types page on [3]www.iana.org for a full reference and explanation of each Media Type.

JSON response data

JSON response structure

All successful responses returned from ComPair Data API Service in JSON format will be in the form of an array. This means that all successful response data in JSON format will begin with the left square bracket `[` (ASCII character code 91) and end with the right square bracket `]` (ASCII character code 93). This also applies to successful returned response data where no records were found. In the case that a request is unsuccessfully processed, the response data will be a single JSON object containing an error code and message. A JSON response containing an error code will start with a left curly bracket `{` (ASCII character code 123) and end with right curly bracket `}` (ASCII character code 125).

Example JSON Error Response

```
{"error": "Bad Request. Request parameters are missing or invalid."}
```

Please see ***Appendix F*** for a full list of response codes and messages available. In most cases, the response headers will contain the error code and it is common practice to inspect the header for such information when handling response data of any type.

For additional information on standard JSON syntax and structure, please visit the [1]json.org.

The below example illustrates a basic request and response sequence where the requested dataset is `services` and the filtering criteria for the dataset is `service_id=4` and `service_name=ACL%20--%20Hapag-Lloyd%2FWallenius%20Wilhelmsen%20-%20A%20Service`:

Example Request

```
https://api.bwrdata.com/get/json?key=<CUSTOMER_API_KEY>&dataset=services&service_id=4&service_name=ACL%20--%20Hapag-Lloyd%2FWallenius%20Wilhelmsen%20-%20A%20Service
```

Note [1]: The "/json" portion of the request indicates the response data should be in JSON format.

Note [2]: The value for `service_name` is URL Encoded in the example request. This is recommended to prevent the inclusion of unsafe characters. Please refer to ***Appendix E*** for additional information about unsafe characters and URL Encoding.

ComPair Data

Example JSON Response

```
[{
      “HEADER”: {
            “RECORDS_RETURNED”: 1
      },
      “DATA”: [{
            "SERVICE_ID": 4,
            "SERVICE_NAME": "ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service",
            "SERVICE_LONG_NME": "ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service
(Transatlantic)",
            "FREQUENCY": "7 days",
            "START_DOW": 4,
            "START_DT": "02/07/2018 00:00",
            "FREQUENCY_IN_DAYS": 7,
            "DIRECTION": "East-West",
            "VESSELDEPLOYMENTAREA": "ECNA-EUR-ECNA"
      }]
}]
```

The above illustrates a response from ComPair Data API Service in JSON format where the criteria parameters included in the request successfully matched data within the ComPair Data system. As mentioned previously, data returned in JSON format will be in the form of an array. Within the outer array is an object containing the data object. The data object contains two properties named “HEADER” and “DATA”. The “HEADER” property contains system information about the response. The “DATA” property contains the matching records based on the submitted request.

Example Empty JSON Response

```
[{
      “HEADER”: {
            “RECORDS_RETURNED”: 0
      },
      “DATA”: []
}]
```

The above illustrates a JSON-formatted response for a request made to ComPair Data’s API Service where no records met the criteria submitted in the request. Within the outer array is a response object containing the two parameters “HEADER” and “DATA”. The HEADER property contains system information about the response. In the example, the information indicates that there are no records returned.

The information included in each “HEADER” property may vary as later versions of ComPair Data’s API service are released. Plans include the addition of statistics information within each successful response object.

Refer to ***Appendix C*** for a complete list of query parameters available for JSON-formatted responses.

Media Type for JSON response data

Each successful response from ComPair Data's API service in JSON format will return in the standard ***.json*** Media Type:

```
application/json
```

For additional information on the [2]application/json Media Type, please refer to [3]iana.org.

HEADER.property

The following is a list of all possible properties available to the "HEADER" property within the response JSON data.

RECORDS_RETURNED	The total number of records returned in the response. This value will be 0 if no records match the request parameters.

Note: The properties for the HEADER property may change in production versions of the API Service.

CSV response data

CSV response structure

Each field value in the response data in CSV format will use double quotes `"` (ASCII character code 34) as the text-qualifier with the default field delimiter being a comma `,` (ASCII character code 44). By default, the text-qualifier will apply to all fields unless the value for that field is null or empty.

Each record in the response data in CSV format will be separated by escape sequence `\n` (ASCII control character code 10) to represent end of line (EOL).

You can optionally request to receive the headers as the first row of data by passing the parameter `include_headers=1` .

The following is an example of data returned in CSV format where dataset was `services` and criteria parameters are `service_id=4` , service_name=ACL%20--%20Hapag-Lloyd%2FWallenius%20Wilhelmsen%20-%20A%20Service and `include_headers=1` :

Example Request:

https://api.bwrdata.com/get/csv?key=<CUSTOMER_API_KEY>&dataset=services&service_id=4&include_headers=1

Example CSV Response:

```
"SERVICE_ID","ALLIANCE_ID","SERVICE_NAME","SERVICE_LONG_NME","FREQUENCY","START_DOW","START_DT","FREQUENCY_IN_DAYS","DIRECTION","VESSELDEPLOYMENTAREA","MODIFIED_DATE","AMENDMENT_CODE"
"4",NULL,"ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service","ACL -- Hapag-Lloyd/Wallenius Wilhelmsen - A Service (Transatlantic)","7 days", "4","02/07/2018 00:00","7","East-West","ECNA-EUR-ECNA","02/08/2018 15:20","U"
```

Unlike JSON formatted responses which return an empty array, CSV formatted responses will return no data if no records are found.

Refer to ***Appendix C*** for a complete list of available query parameters.

Media Type for CSV response data

Each successful response from ComPair Data's API Service in CSV format will return in the standard text Media Type:

```
text/plain
```

Filters & Features

Single value filtering

Filtering an API request can be accomplished by supplying the name of the field within the request followed by = and then the value by which to filter – just like any other query parameter. The field name must be in all lowercase and no quotes are required when supplying the value.

Syntax	Example	Result
`field_name=field_value`	`service_id=4`	All data where service_id is 4.
`field_name1_field_value1&field_name2=field_value2`	`service_id=4&start_dt=02/07/2018 00:00`	All data where the service_id is 4

		and the start_dt is 02/07/2018 00:00.

Multi-value filtering

To submit an API request where the filtering criteria indicates multiple values, supply the field name followed by = just as with a single value parameter. Following the = is, instead, a JSON array containing all values by which to filter the data.

Syntax	Example	Result
field_name=[field_value1,field_value2]	service_id=[4,26]	All data where the service id is 4 or 26.
field_name=[field_value1,field_value2]	service_type=[“Fully Cellular”,”Ro-Ro”]	All data where the service_type is Fully Cellular or Ro-Ro.

Note that quotes are required for text values. Numerical values may have the quotes omitted in the request.

Wildcard value filtering

The use of wildcards will allow for filtering based on an approximate value. Currently only single value filtering supports the use of wildcards. The wildcard character `*` represents any character (including spaces, tabs, and even line breaks).

For this example, we will be searching through the `vessels` dataset, where fields such as the IMO number (`vessel_code`) and the Vessel Name (`vessel_nme`) are available.

Syntax	Example	Result
field_name=*field_value	vessel_nme=*Maersk	All data where the vessel_nme field value ends with “Maersk”.

field_name=field_value*	vessel_nme=Maersk*	All data where the vessel_nme field value starts with “Maersk”.
field_name=*field_value*	vessel_nme=*Maersk*	All data where the vessel_nme field value contains “Maersk”.

Large datasets & Pagination

When querying larger datasets or sending requests to ComPair Data’s API Service with a high limit value, it is recommended to take advantage of pagination. This method uses the `offset` and `limit` parameters within the request to rapidly traverse over thousands of records within a dataset. As an example, to query the first 3 pages of a large dataset of 245,000 records, ***Figure 3*** shows three separate requests for each of the pages of data.

Figure 3

Pages	API Request	Result
1	…/get/csv?key=<myAPIkey>&dataset=<myLargeDataset>&limit=100000	Returns first 100000 records.
2	…/get/csv?key=<myAPIkey>&dataset=<myLargeDataset>&limit=100000&offset=100000	Returns second 100000 records.
3	…/get/csv?key=<myAPIkey>&dataset=<myLargeDataset>&limit=100000&offset=200000	Returns third 100000 records. Since there are fewer than 100000 on page 3,

		returns only the remaining 45,000.

Attempting to query for page 4 will return no data for CSV format, and a value of `0` in the RECORDS_RETURNED property in JSON format.

Date Ranges

For some datasets, it's important to limit the scope of the data by a range of dates. For example, in the voyageschedules dataset, you may wish to filter by port calls between date A and date B, or all port calls greater than or less than date A.

The syntax for date ranges works just like a regular parameter, with the name of the field followed by = and then the value by which to filter. To initiate a range of dates, a dash (-) is used to indicate "dates from …" and "dates to …".

Syntax	Example	Result
MM/DD/YYYY-MM/DD/YYYY	09/20/2018-09/27/2018	All data where the date is between or equal to 09/20/2018 and 09/27/2018.
MM/DD/YYYY-	09/20/2018-	All data where the date is greater than or equal to 09/20/2018.
-MM/DD/YYYY	-09/27/2018	All data where the date is less than or equal to 09/27/2018.

Refer to ***Appendix C*** for a complete list of available query parameters.

API Request Limits

The limits assigned to API keys depends on the level of service to which the customer has subscribed and indicates the number of successful API request that can be made at any given time. If the API Request Limit is set to 1000 requests per month, this means that for each successful request made to ComPair Data's API Service, the usage count will only increment by a count of one request for each successful request, regardless of the amount of data requested. Requests totaling over the 1000 request-limit for the month will fail. Limits of this type depend on the subscription, such as a subscription for unlimited access.

The number of records returned by the request does not factor into the API Request Limit -- however, there are restrictions on the size of all response data. Any request in which the size of the response from ComPair Data's API Service is larger than approximately 524,288KB (a little over half a gigabyte) will likely fail. In these scenarios, pagination is recommended.

Dataset Restrictions

The datasets available to each API request is limited to the datasets to which the customer has subscribed. If the customer has not subscribed to the **P2PSchedules** dataset, for example, any requests made to this dataset will fail with error code 401. The number of datasets allowed per customer depends on the level of service to which the customer has subscribed. Each dataset has its own set of fields and field descriptions are available upon request.

Custom Datasets

At request, the API Service users can subscribe to custom datasets – the specifications of which are defined by the subscriber. Customization may include the name of the dataset, the addition of other fields, the formatting of certain fields, the pre-defined scope of the data, or the availability of datasets not offered as part of the standard list of datasets. Depending on the complexity of the custom dataset, customization may incur additional charges relative to the development effort required. If the dataset becomes extremely large, the subscriber will have the option of a dedicated server at an additional cost. Please contact edi-support@shippers.com for a quote on customization requirements.

Technical, Service Level, Dataset Questions

For technical questions or questions related to service levels, available datasets, and service limitations, contact ComPair Data at edi-support@shippers.com

Appendix

Appendix A

Client Application Usage Example:

Successful requests made by an application on a client's system may access response data in JSON format as in the following ***Javascript*** example:

Javascript Example:

```
var APIRequest = function(requestURL, callback) {
      /* logic for the application making the request */
```

```
};

APIRequest("https://api.bwrdata.com/get/json?key=<CUSTOMER_API_KEY>&dataset=services&service_id=4", (res)=> {
	var Response;
	try {

		Response = JSON.parse(res);

		if(Array.isArray(Response)) {

			console.log(Response[0].HEADER.RECORDS_RETURNED);
			/* Will output "1” to the console. */

			console.log(Response[0].DATA[0].FREQUENCY);
			/* Will output "7 days" to the console. */


	console.log(Response[0].DATA[0].VESSELDEPLOYMENTAREA);
			/* Will output "ECNA-EUR-ECNA" to the console. */

		} else {
			/* logic for handling API service errors. */
		}


	} catch(error) {
		/* handle unexpected errors */
	}
});
```

Appendix B

Amendment Code definitions:

A	**ADD**	Indicates that the data is new. If the record already exists within the client’s system, the data should be ignored.
U	**UPDATE**	Indicates that the data has been updated. The accompanying MODIFIED_DATE field will indicate the date of the update and can be used to compare with existing data in the client’s system to determine if an overwrite is required, otherwise the data should be ignored.
D	**DELETE**	Indicates that this data is to be deleted from the client’s system and that this data has been deleted from the ComPair Data system.

ComPair Data

Appendix C

Query String Parameters

The following is a list of available query string parameters used to modify the response data.

Parameter	Description	Example
limit	A number value that restricts the number of records returned in the response.	limit=100
offset	A number value that jumps to the nth record of a dataset and returns the data from that record onward. Typically used for pagination of data.	offset=100
sort	A comma separated list of fields by which to sort the response data. Adding a "-" (minus sign) in front of a field name will sort that field in descending order.	sort=service_id,-modified_date
pretty	Setting this parameter to a value of 1 will render JSON response data in a way that is easier to read by the human eye. See ***Appendix D*** for a comparison between pretty, and non-pretty settings.	pretty=1
dataset	The name of the dataset from which to request data. Some datasets may not be available to all customers.	dataset=vessels
key	The private API key used by the customer to access API Services. This is required with all requests.	key=XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Appendix D

JSON Response in "Pretty" Mode

Without pretty=1


```
[{"HEADER":{"RECORDS_RETURNED":1},"DATA":[{"SERVICESCHEDULE_MASTERVOYAGE_ID":"61118648803
","VOYAGE_ID":"24293","GENERATEDSERVICEID":"MSCU/9235610/WT817R","CARRIER_SERVICE_DESIGNA
TION":"LIBREVILLE FEEDER
SERVICE","CARRIERCODE":"MSCU","CARRIERNAME":"MSC","COMPAIRSERVICESHORTNAME":null,"FREQUEN
CY":null,"VOYAGESTARTDOW":"Friday","VESSELCODE":"9235610","VESSELNAME":"CONTSHIP
GEM","VESSELFLAG":null,"TEUCAPACITY":null,"REEFERPLUGS":null,"VESSELVOYAGENUMBER":"WT817R
","PORTCODE":"MYPKG","PORTNAME":"Port
Kelang","TERMINALNAME":null,"LOCATIONTYPECODE":"D","AMENDMENTCODE":"U","EVENTCODECOMPAIR"
:"EAD","EVENTDATE":"06/11/2018 13:00","MODIFIED_DATE":"04/23/2018
20:00","CARRIER_ID":"73","SERVICE_ID":null,"REMARKS":"LFP-GALBV"}]}]
```


Note: The data in the above table is rendered in a single line with no line breaks or indentation.

With pretty=1


```
[
        {
                "HEADER": {
                        "RECORDS_RETURNED": 1
                },
                "DATA": [
                        {
                                "SERVICESCHEDULE_MASTERVOYAGE_ID": "61118648803",
                                "VOYAGE_ID": "24293",
                                "GENERATEDSERVICEID": "MSCU/9235610/WT817R",
                                "CARRIER_SERVICE_DESIGNATION": "LIBREVILLE FEEDER SERVICE",
                                "CARRIERCODE": "MSCU",
                                "CARRIERNAME": "MSC",
                                "COMPAIRSERVICESHORTNAME": null,
                                "FREQUENCY": null,
                                "VOYAGESTARTDOW": "Friday",
                                "VESSELCODE": "9235610",
                                "VESSELNAME": "CONTSHIP GEM",
                                "VESSELFLAG": null,
                                "TEUCAPACITY": null,
                                "REEFERPLUGS": null,
                                "VESSELVOYAGENUMBER": "WT817R",
                                "PORTCODE": "MYPKG",
                                "PORTNAME": "Port Kelang",
                                "TERMINALNAME": null,
                                "LOCATIONTYPECODE": "D",
                                "AMENDMENTCODE": "U",
                                "EVENTCODECOMPAIR": "EAD",
                                "EVENTDATE": "06/11/2018 13:00",
                                "MODIFIED_DATE": "04/23/2018 20:00",
                                "CARRIER_ID": "73",
                                "SERVICE_ID": null,
                                "REMARKS": "LFP-GALBV"
                        }
                ]
        }
]
```


Appendix E

Unsafe characters in API Request

The following list of characters should not be included as the name or value of a query parameter. Using such characters will increase the chances of a failed or bad request error or undesired response data.

Unsafe Character	Character Description
;	semicolon
/	forward slash

?	question mark
:	colon
@	at
=	equal
&	ampersand

Example unsafe request #1:

https://api.bwrdata.com/get/json?key=X00073BA-0000-FE43-BE00-84000X5CD0XD&dataset=services&service_nme=**Caronila & Company Express/CE2**

This example demonstrates a request containing both an ampersand (&) and a forward slash (/) within the value of the query parameter (bolded text). Due to the unsafe characters, this request will result in either an error or produce undesired response data. In this scenario, it is recommended to URL Encode any values to ensure they are not interpreted as unsafe characters.

When the value in example #1 is URL Encoded, it will look like the following:

Caronila%20%26%20Company%20Express%2FCE2

When the API Service receives this value, it will interpret the value as:

Caronila & Company Express/CE2

Many programming languages contain a built-in URL Encoding method. A value can be manually URL Encoded using sites such as [4]urlencode.org which is an input/output web application that encodes and decodes text.

Example unsafe request #2:

https://api.bwrdata.com/get/json?key=X00073BA-0000-FE43-BE00-84000X5CD0XD&dataset=services&**service_@_short_@_name**=Caronila%20%26%20Company%20Express%2FCE2

Appendix F

Error Response Codes

Error Code	Error Description
400	Bad Request. Request parameters are missing or invalid.

401	Unauthorized. The API key is invalid. (Usually an account or permissions related issue).
404	Page not found.
408	Request timeout. Server is busy and could not process your request. Please try again later.
429	Too many requests. Please try again later.
500	Something went wrong. An internal server error has occurred.

Appendix G

API Keys

API Keys are private authorization codes that permit requests to the ComPair Data API Service. Only one API Key can be issued per license, per customer, and it is recommended that this key not be shared, or otherwise risk potential unwanted usage charges or account limits being reached without warning. A customer can have the option of having multiple licenses, and therefore multiple API keys, each with its own level of access.

If an API Key is lost, a new one may be requested. Upon receiving the new API Key, the old API Key will be deauthorized. Please keep this in mind when developing applications or systems around an API Key.

Any unauthorized API Keys will receive a 401 error when attempting to make requests to the ComPair Data API Service. If you receive a 401 error, it could mean any one of the following scenarios:

A. The API Key has expired.
B. The API Key is invalid or incorrect.
C. The API Key has been deauthorized.
D. The API Key is valid, but access to a dataset is not authorized for this license.

References

1. JSON.org – Introducing JSON (https://www.json.org)
2. JSON Media Type (https://www.iana.org/assignments/media-types/application/json)
3. IANA.org – List of Media Types (http://www.iana.org/assignments/media-types/media-types.xhtml)
4. URLEncode.org (https://urlencode.org)