

# Roller API Data Model

*for use with Invantive SQL*



# Copyright

(C) Copyright 2004-2023 Invantive Software B.V., the Netherlands. All rights reserved.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Despite all the care taken in the compilation of this text, neither the author nor the publisher can accept liability for any damage, which might result from any error, which might appear in this publication.

This manual is a reference guide intended to clarify usage. If data in the sample images match data in your system, the similarity is coincidental.

## Important Safety and Usage Information

**Intended Use and Limitations:** This software, developed by Invantive, is designed to support a variety of business and information technology data processing functions, such as accounting, financial reporting and sales reporting. It is important to note that this software is not designed, tested, or approved for use in environments where malfunction or failure could lead to life-threatening situations or severe physical or environmental damage. This includes, but is not limited to:

- Nuclear facilities: The software should not be used for operations or functions related to the control, maintenance, or operation of nuclear facilities.
- Defense and Military Applications: This software is not suitable for use in defense-related applications, including but not limited to weaponry control, military strategy planning, or any other aspects of national defense.
- Aviation: The software is not intended for use in the operation, navigation, or communication systems of any aircraft or air traffic control environments.
- Healthcare and Medicine Production: This software should not be utilized for medical device operation, patient data analysis for critical health decisions, pharmaceutical production, or medical research where its failure or malfunction could impact patient health.
- Chemical and Hazardous Material Handling: This software is not intended for the management, control, or operational aspects of chemical plants or hazardous material handling facilities. Any malfunction in software used in these settings could result in dangerous chemical spills, explosions, or environmental disasters.
- Transportation and Traffic Control Systems: The software should not be used for the control, operation, or management of transportation systems, including railway signal controls, subway systems, or traffic light management. Malfunctions in such critical systems could lead to severe accidents and endanger public safety.
- Energy Grid and Utility Control Systems: This software is not designed for the control or operation of energy grid systems, including electrical substations, renewable energy control systems, or water utility control systems. The failure of software in these areas could lead to significant power outages, water supply disruptions, or other public utility failures, potentially endangering communities and causing extensive damage.
- Other High-Risk Environments: Any other critical infrastructure and environments where a failure of the software could result in significant harm to individuals or the environment.

**User Responsibility:** Users must ensure that they understand the intended use of the software and refrain from deploying it in any setting that falls outside of its designed purpose. It is the responsibility of the user to assess the suitability of the software for their intended application, especially in any scenarios that might pose a risk to life, health, or the environment.

**Disclaimer of Liability:** Invantive disclaims any responsibility for damage, injury, or legal consequences resulting from the use or misuse of this software in prohibited or unintended applications.

# Contents

<b>1</b>	<b>SQL Driver for Roller API</b>	<b>1</b>
<b>2</b>	<b>SQL Driver Attributes for Roller API</b>	<b>2</b>
<b>3</b>	<b>Schema: Data</b>	<b>15</b>
<b>3.1</b>	<b>Tables</b> .....	<b>15</b>
3.1.1	AttendanceLocationsByDate: Roller Attendance Location by Date .....	15
3.1.2	AttendancesByDate: Roller Attendances by Date .....	16
3.1.3	BookingById: Roller Booking by ID .....	17
3.1.4	BookingItems: Roller Booking Items .....	19
3.1.5	BookingItemsByBookingId: Roller Booking Items by ID .....	21
3.1.6	BookingItemsByDate: Roller Booking Items by Date .....	22
3.1.7	BookingPaymentsByDate: Roller Booking Payments by Date .....	24
3.1.8	Bookings: Roller Bookings .....	26
3.1.9	BookingSignedWaiversByDate: Roller Booking Signed Waivers by Date .....	27
3.1.10	CustomerById: Roller Customer by ID .....	28
3.1.11	CustomersByDate: Roller Customers by Date .....	30
3.1.12	Devices .....	31
3.1.13	DiscountCodes: Roller Discount Codes .....	32
3.1.14	DiscountProducts: Roller Discount Products .....	33
3.1.15	Discounts: Roller Discounts .....	35
3.1.16	GiftCardsByDate: Roller Gift Cards by Date .....	36
3.1.17	Locations: Roller Locations .....	38
3.1.18	MembershipCreditsByDate: Roller Membership Statuses .....	38
3.1.19	MembershipRedemptionsByDate: Roller Membership Redemptions .....	40
3.1.20	MembershipStatusesByDate: Roller Membership Statuses .....	41
3.1.21	Modifiers .....	42
3.1.22	ProductAvailabilitiesByDate: Roller Product Availabilities by Date .....	42
3.1.23	ProductAvailabilityAllocationsByDate: Roller Product Availability Product Session Allocations by Date ....	44
3.1.24	ProductAvailabilityByDate: Roller Product Availability by Date .....	45
3.1.25	ProductAvailabilityProductsByDate .....	46
3.1.26	ProductAvailabilitySessionAllocationsByDate: Roller Product Availability Product Session Allocations by Date	
3.1.27	ProductAvailabilitySessionsByDate: Roller Product Availability Product Sessions by Date .....	49
3.1.28	Products: Roller Products .....	50
3.1.29	ReportingCategories: Roller Reporting Categories .....	51
3.1.30	ReportingCategoryCategories: Roller Reporting Categories .....	52
3.1.31	ReportingCategoryProducts: Roller Reporting Products .....	52
3.1.32	Revenues: Roller Revenues .....	53
3.1.33	SignedWaiversByDate: Roller Signed Waivers by Date .....	55
3.1.34	StaffMembers: Roller Staff Members .....	57
3.1.35	TicketDiscountsByDate: Roller Ticket Discounts by Date .....	58
3.1.36	TicketsByDate: Roller Tickets by Date .....	59
3.1.37	TillReconciliations .....	61
3.1.38	Venues .....	62
3.1.39	Waivers: Roller Waivers .....	63
3.1.40	Webhooks .....	64
<b>4</b>	<b>Schema: Native</b>	<b>65</b>
<b>4.1</b>	<b>Tables</b> .....	<b>65</b>
4.1.1	NATIVEPLATFORMSCALARREQUESTS: Roller Native Platform Scalar Requests .....	65

# Index

**67**

# 1 SQL Driver for Roller API

Invantive SQL is the fastest, easiest and most reliable way to exchange data with the Roller API.

Use the "Search" option in the left menu to search for a specific term such as the table or column description. When you already know the term, please use the "Index" option. When you can't find the information needed, please click on the Chat button at the bottom or place your question in the [user community](#). Other users or Invantive Support will try to help you to our best.

Online software for attractions, entertainment and leisure venues.

The Roller driver covers 41 tables and 452 columns.

## Roller API Clients

Invantive SQL is available on many user interfaces ("clients" in traditional server-client paradigm). All Invantive SQL statements can be exchanged with a close to 100% compatibility across all clients and operating systems (Windows, MacOS, Linux, iOS, Android).

The clients include Microsoft Excel, Microsoft Power BI, Microsoft Power Query, Microsoft Word and Microsoft Outlook. Web-based clients include Invantive Cloud, Invantive Bridge Online as OData proxy, Invantive App Online for interactive apps, Online SQL Editor for query execution and Invantive Data Access Point as extended proxy.

The [Roller Power BI connector](#) is based on the Invantive SQL driver for Roller, completed by a high-performance OData connector which works straight on Power BI without any add-on. The OData protocol is always version 4, independent whether the backing platform uses OData, SOAP or another protocol.

For technical users there are command-line editions of Invantive Data Hub running on iOS, Android, Windows, MacOS and Linux. Invantive Data Hub is also often used for enterprise server applications such as ETL. High-volume replication of data taken from the Roller API into traditional databases such as SQL Server (on-premise and Azure), MySQL, PostgreSQL and Oracle is possible using [Invantive Data Replicator](#). Invantive Data Replicator automatically creates and maintains Roller datawarehouses, possibly in combination with data from over 70 other (cloud) platforms. Data Replicator supports data volumes up to over 1 TB and over 5.000 companies. The on-premise edition of Invantive Bridge offers an Roller ADO.net provider.

Finally, online web apps can be build for Roller using App Online of [Invantive Cloud](#).

## Monitor API Calls

When a query or DML-statement has been executed on Invantive SQL a developer can evaluate the actual calls made to the Roller API using a query on sessionios@DataDictionary. As an alternative, extensive request and response logging can be enabled by setting log-native-calls-to-disk to true. In the %USERPROFILE%\Invantive\NativeLog folder Invantive SQL will create log files per API request and response.

## Specifications

The SQL driver for Roller does not support partitioning. Define one data container in a database for each company in Roller to enable parallel access for data from multiple companies.

An introduction into the concepts of Invantive SQL such as databases, data containers and partitioning can be found in the [Invantive SQL grammar](#).

The configuration can be changed using various attributes during log on and use. A full list of configuration options is listed in the [driver attributes](#) [2].

The catalog name is used to compose the full qualified name of an object like a table or view. The schema name is used to compose the full qualified name of an object like a table or view. On Roller the comparison of two texts is case sensitive by default.

Changes and bug fixes on the Roller SQL driver can be found in the [release notes](#). Get access to the Roller community through the [Roller section](#) of the Invantive forums.

Driver code for use in settings.xml: `Roller`

Alias: `roller`

Recommended alias: `rlr`

More technical documentation as provided by the supplier of the Roller API on the native APIconnection used can be found at <https://docs.roller.app/>.

General documentation on Roller is available at <https://roller.software>

The Roller software for attractions requires additional purchase of API calls. The monthly allowance starts at 5.000 API calls per month. In general it is recommended to use the table functions with varying dates to load the data incrementally without using all API calls in a short time window.

Updated: 22-09-2022 21:51 using Invantive SQL version 22.0.355-PROD+3587.

## 2 SQL Driver Attributes for Roller API

The SQL driver for Roller has many attributes that can be finetuned to improve handling in scenarios with unreliable network connections to the API server of Roller or high-volumes of data. Also, many drivers have driver-specific attributes to finetune actual behaviour or handle data not matching specifications.

The Roller driver attributes are assigned a default value which seldom requires change. However, changes can be applied when needed on four levels, which are reflected in the table below by separate checkmarks:

- Connection string: the connection string from the settings\*.xml file and applied during log on.
- Set SQL statement: a set SQL-statement to be executed once connection has been established.
- Drivers file: the providers.xml file (obsolete starting release 17.32).
- Log on: value to be specified interactively by user during log on in a user interface.

The connection string for Roller can be found in the settings\*.xml file used for the database. Settings\*.xml files are typically located in the %USERPROFILE%\invantive folder in most deployment scenarios. The reference manuals contain instructions how to relocate the settings\*.xml files. Each data container of a database in the connection string can have a `connectionString` element specifying the name and values of attributes. Both name and value must be properly escaped according to XML-semantics. Actual application of the value is solely done during log on. A new connection must be established to change the value of a driver attribute using a connection string.

The set SQL statement can be executed after log on. The syntax is: `set NAME VALUE`, or for a distributed database: `set NAME@ALIAS VALUE`. In some scenarios you may need to enclose the driver attribute name in square brackets to escape it from parsing, for instance when a reserved SQL keyword is part of the name. The new value takes effect straight after execution of the set-statement. The set-statement can be executed as often as needed during a session.

Driver attributes that can be interactively set to a value are typically presented in the log on window. Depending on the platform and design decisions of the user interface designer, some or all of the available driver attributes can have been made available.

The Roller driver can be configured using the following attributes:

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
add-odata-mandatory-filters	Whether to automatically add OData filters deemed necessary by the platform.	OData	False	✓	✓	✓	
analysis-enforce-row-uniqueness	Use for analysis only! Enforce rows to be unique.	Shared	False	✓	✓	✓	
api-access-token	Access Token is a security token for multiple OAuth2 Flows. With an Access Token you can access protected resources. An Access Token must be stored securely since once compromised allows access to your protected resources.	OData		✓		✓	✓
api-client-id	The client ID is a unique identifier of your application. It is generated by registering an application.	OData		✓		✓	✓
api-client-secret	The client secret is to be kept confidential. Such as a password for a logon code, the client secret is the confidential part of an app identified by a client ID. It is needed during the OAuth2 Code Grant Flow together with the refresh token to get access.	OData		✓		✓	✓
api-pre-expiry-refresh-sec	The number of seconds before the token expires to acquire a new token.	OData		✓	✓	✓	
api-redirect-url	The redirect URI is the website a browser session is redirected to after the OAuth2 authentication process has been completed.	OData		✓		✓	✓
api-refresh-token	Refresh Token is a security token for the OAuth2 Code Grant Flow. With a Refresh Token and client secret you can retrieve a renewed access token to access protected resources. A Refresh Token and client secret must be stored securely since once compromised allows access to your protected resources.	OData		✓		✓	✓

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
api-scope	The scope to request an OAuth token for.	OData		✓		✓	
api-token-url	The token URI is the OAuth2 endpoint to exchange tokens.	OData		✓		✓	
api-url	URL to access the API.	OData		✓		✓	
bulk-delete-page-size-rows	Number of rows to delete per batch when bulk deleting	Shared	10000	✓	✓	✓	
bulk-insert-page-size-bytes	Approximate maximum size in bytes of batch when bulk inserting	Shared	10000000	✓	✓	✓	
bulk-insert-page-size-rows	Number of rows to insert per batch when bulk inserting	Shared	250	✓	✓	✓	
download-error-400-bad-request-max-tries	Maximum number of tries when OData server reports bad format during retrieval of data.		3	✓	✓	✓	
download-error-400-bad-request-sleep-initial-ms	Initial sleep in milliseconds between retries when OData server reports that the API server is unavailable during retrieval of data.		500	✓	✓	✓	
download-error-400-bad-request-sleep-max-ms	Maximum sleep in milliseconds between retries when OData server reports that the API server is unavailable during retrieval of data.		5000	✓	✓	✓	
download-error-400-bad-request-sleep-multiplicator	Multiplication factor for sleep between retries OData server reports that the API server is unavailable during retrieval of data.		2	✓	✓	✓	
download-error-408-request-timeout-max-tries	Maximum number of tries when the website reports a HTTP status 408.		10	✓	✓	✓	
download-error-408-request-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 408.		10000	✓	✓	✓	
download-error-408-request-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 408.		300000	✓	✓	✓	
download-error-408-request-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 408.		2	✓	✓	✓	
download-error-422-bad-request-max-tries	Maximum number of tries when OData server reports unprocessable entity during retrieval of data.		30	✓	✓	✓	
download-error-422-bad-request-sleep-initial-ms	Initial sleep in milliseconds between retries when OData server reports unprocessable entity during retrieval of data.		10000	✓	✓	✓	
download-error-422-bad-request-sleep-max-ms	Maximum sleep in milliseconds between retries when OData server reports unprocessable entity during retrieval of data.		300000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
dow nload-error-422-bad-request-sleep-multiplicator	Multiplication factor for sleep between retries OData server reports unprocessable entity during retrieval of data.		2	✓	✓	✓	
dow nload-error-429-too-many-requests-max-tries	Maximum number of tries when the website reports that too many requests have been made during a timeslot of one minute or one day.		10	✓	✓	✓	
dow nload-error-429-too-many-requests-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports that too many requests have been made during a timeslot of one minute or one day.		10000	✓	✓	✓	
dow nload-error-429-too-many-requests-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports that too many requests have been made during a timeslot of one minute or one day.		300000	✓	✓	✓	
dow nload-error-429-too-many-requests-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports that too many requests have been made during a timeslot of one minute or one day.		2	✓	✓	✓	
dow nload-error-502-server-unavailable-max-tries	Maximum number of tries when OData server reports a bad gateway during retrieval of data.		30	✓	✓	✓	
dow nload-error-502-server-unavailable-sleep-initial-ms	Initial sleep in milliseconds between retries when OData server reports a bad gateway during retrieval of data.		10000	✓	✓	✓	
dow nload-error-502-server-unavailable-sleep-max-ms	Maximum sleep in milliseconds between retries when OData server reports that a bad gateway during retrieval of data.		300000	✓	✓	✓	
dow nload-error-502-server-unavailable-sleep-multiplicator	Multiplication factor for sleep between retries OData server reports a bad gateway during retrieval of data.		2	✓	✓	✓	
dow nload-error-503-server-unavailable-max-tries	Maximum number of tries when OData server reports that the API server is unavailable during retrieval of data.		30	✓	✓	✓	
dow nload-error-503-server-unavailable-sleep-initial-ms	Initial sleep in milliseconds between retries when OData server reports that the API server is unavailable during retrieval of data.		10000	✓	✓	✓	
dow nload-error-503-server-unavailable-sleep-max-ms	Maximum sleep in milliseconds between retries when OData server reports that the API server is unavailable during retrieval of data.		300000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
dow nload-error-503-server-unavailable-sleep-multiplicator	Multiplication factor for sleep between retries OData server reports that the API server is unavailable during retrieval of data.		2	✓	✓	✓	
dow nload-error-504-gateway-timeout-max-tries	Maximum number of tries when the website reports a gateway timeout.		10	✓	✓	✓	
dow nload-error-504-gateway-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a gateway timeout.		10000	✓	✓	✓	
dow nload-error-504-gateway-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a gateway timeout.		300000	✓	✓	✓	
dow nload-error-504-gateway-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a gateway timeout.		2	✓	✓	✓	
dow nload-error-590-netw ork-connect-timeout-max-tries	Maximum number of tries when the website reports a HTTP status 590.		10	✓	✓	✓	
dow nload-error-590-netw ork-connect-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 590.		10000	✓	✓	✓	
dow nload-error-590-netw ork-connect-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 590.		300000	✓	✓	✓	
dow nload-error-590-netw ork-connect-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 590.		2	✓	✓	✓	
dow nload-error-599-netw ork-connect-timeout-max-tries	Maximum number of tries when the website reports a HTTP status 599.		10	✓	✓	✓	
dow nload-error-599-netw ork-connect-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 599.		10000	✓	✓	✓	
dow nload-error-599-netw ork-connect-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 599.		300000	✓	✓	✓	
dow nload-error-599-netw ork-connect-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 599.		2	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
download-error-argument-exception-max-tries	Maximum number of tries when an argument exception is returned when downloading a blob.		10	✓	✓	✓	
download-error-argument-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when an argument exception is returned when downloading a blob.		10000	✓	✓	✓	
download-error-argument-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when an argument exception is returned when downloading a blob.		300000	✓	✓	✓	
download-error-argument-exception-sleep-multiplicator	Multiplication factor for sleep between retries when an argument exception is returned when downloading a blob.		2	✓	✓	✓	
download-error-internet-download-max-tries	Maximum number of tries when the Internet connection seems down during retrieval of data.		10	✓	✓	✓	
download-error-internet-download-sleep-initial-ms	Initial sleep in milliseconds between retries when the Internet connection seems down during retrieval of data.		10000	✓	✓	✓	
download-error-internet-download-sleep-max-ms	Maximum sleep in milliseconds between retries when the Internet connection seems down during retrieval of data.		300000	✓	✓	✓	
download-error-internet-download-sleep-multiplicator	Multiplication factor for sleep between retries when the Internet connection seems down during retrieval of data.		2	✓	✓	✓	
download-error-io-exception-max-tries	Maximum number of tries when a network I/O connection failure occurs during retrieval of data.		10	✓	✓	✓	
download-error-io-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when a network I/O connection failure occurs during retrieval of data.		10000	✓	✓	✓	
download-error-io-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when a network I/O connection failure occurs during retrieval of data.		300000	✓	✓	✓	
download-error-io-exception-sleep-multiplicator	Multiplication factor for sleep between retries when a network I/O connection failure occurs during retrieval of data.		2	✓	✓	✓	
download-error-json-exception-max-tries	Maximum number of tries when an invalid JSON body is returned.		3	✓	✓	✓	
download-error-json-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when an invalid JSON body is returned.		1000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
dow nload-error- json-exception- sleep-max-ms	Maximum sleep in milliseconds between retries when an invalid JSON body is returned.		10000	✓	✓	✓	
dow nload-error- json-exception- sleep-multiplicator	Multiplication factor for sleep between retries when an invalid JSON body is returned.		2	✓	✓	✓	
dow nload-error- other-exception- max-tries	Maximum number of tries when an unqualified error occurs during retrieval of data.		3	✓	✓	✓	
dow nload-error- other-exception- sleep-initial-ms	Initial sleep in milliseconds between retries when an unqualified error occurs during retrieval of data.		10000	✓	✓	✓	
dow nload-error- other-exception- sleep-max-ms	Maximum sleep in milliseconds between retries when an unqualified error occurs during retrieval of data.		300000	✓	✓	✓	
dow nload-error- other-exception- sleep-multiplicator	Multiplication factor for sleep between retries when an unqualified error occurs during retrieval of data.		2	✓	✓	✓	
dow nload-error- socket-exception- max-tries	Maximum number of tries when the network connection is forcibly dropped during retrieval of data.		10	✓	✓	✓	
dow nload-error- socket-exception- sleep-initial-ms	Initial sleep in milliseconds between retries when the network connection is forcibly dropped during retrieval of data.		10000	✓	✓	✓	
dow nload-error- socket-exception- sleep-max-ms	Maximum sleep in milliseconds between retries when the network connection is forcibly dropped during retrieval of data.		300000	✓	✓	✓	
dow nload-error- socket-exception- sleep-multiplicator	Multiplication factor for sleep between retries when the network connection is forcibly dropped during retrieval of data.		2	✓	✓	✓	
dow nload-error- web-exception-max- tries	Maximum number of tries when a web connection failure occurs during retrieval of data.		10	✓	✓	✓	
dow nload-error- web-exception- sleep-initial-ms	Initial sleep in milliseconds between retries when a web connection failure occurs during retrieval of data.		10000	✓	✓	✓	
dow nload-error- web-exception- sleep-max-ms	Maximum sleep in milliseconds between retries when a web connection failure occurs during retrieval of data.		300000	✓	✓	✓	
dow nload-error- web-exception- sleep-multiplicator	Multiplication factor for sleep between retries when a web connection failure occurs during retrieval of data.		2	✓	✓	✓	
dow nload-error- web-not-	Maximum number of tries when the connection reports not implemented.		1	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
implemented-max-tries							
download-error-web-not-implemented-sleep-initial-ms	Initial sleep in milliseconds between retries when the connection reports not implemented.		10000	✓	✓	✓	
download-error-web-not-implemented-sleep-max-ms	Maximum sleep in milliseconds between retries when the connection reports not implemented.		300000	✓	✓	✓	
download-error-web-not-implemented-sleep-multiplicator	Multiplication factor for sleep between retries when the connection reports not implemented.		2	✓	✓	✓	
download-error-web-timeout-max-tries	Maximum number of tries when the connection reports a timeout.		10	✓	✓	✓	
download-error-web-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the connection reports a timeout.		1000	✓	✓	✓	
download-error-web-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the connection reports a timeout.		30000	✓	✓	✓	
download-error-web-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the connection reports a timeout.		2	✓	✓	✓	
download-error-web-unauthorized-max-tries	Maximum number of tries when the connection reports an unauthorized error.		1	✓	✓	✓	
download-error-web-unauthorized-sleep-initial-ms	Initial sleep in milliseconds between retries when the connection reports an unauthorized error.		10000	✓	✓	✓	
download-error-web-unauthorized-sleep-max-ms	Maximum sleep in milliseconds between retries when the connection reports an unauthorized error.		300000	✓	✓	✓	
download-error-web-unauthorized-sleep-multiplicator	Multiplication factor for sleep between retries when the connection reports an unauthorized error.		2	✓	✓	✓	
force-case-sensitive-identifiers	Consider identifiers as case-sensitive independent of the platform capabilities.	Shared	False	✓	✓	✓	
forced-casing-identifiers	Forced casing of identifiers. Choose from Unset, Lower, Upper and Mixed.	Shared		✓	✓	✓	
http-disk-cache-compression-level	Compression level for the HTTP disk cache, ranging from 1 (little) to 9 (intense). Default is 5.	Shared	5	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
http-disk-cache-directory	Directory where HTTP cache is stored.	Shared	C:\Users\gle3.WS212\Invasive\Cache\http\gle3\shared	✓	✓	✓	
http-disk-cache-ignore-write-errors	Whether to ignore write errors to disk cache.	Shared	False	✓	✓	✓	
http-disk-cache-max-age-sec	Maximum acceptable age in seconds for use of data in the HTTP disk cache.	Shared	2592000	✓	✓	✓	
http-get-timeout-max-ms	HTTP GET maximum timeout on retry (ms).		300000	✓	✓	✓	
http-get-timeout-ms	HTTP GET timeout (ms).		60000	✓	✓	✓	
http-memory-cache-compression-level	Compression level for the HTTP memory cache, ranging from 1 (little) to 9 (intense). Default is 5.	OData	5	✓	✓	✓	
http-memory-cache-max-age-sec	Maximum acceptable age in seconds for use of data in the HTTP memory cache.	OData	14400	✓	✓	✓	
http-post-timeout-max-ms	HTTP POST maximum timeout on retry (ms).		300000	✓	✓	✓	
http-post-timeout-ms	HTTP POST timeout (ms).		300000	✓	✓	✓	
ignore-http-400-errors	Ignore HTTP 400 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-401-errors	Ignore HTTP 401 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-402-errors	Ignore HTTP 402 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-403-errors	Ignore HTTP 403 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-404-errors	Ignore HTTP 404 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-422-errors	Ignore HTTP 422 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-429-errors	Ignore HTTP 429 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-500-errors	Ignore HTTP 500 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
ignore-http-502-errors	Ignore HTTP 502 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
ignore-http-503-errors	Ignore HTTP 503 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
invalid-json-on-get-max-tries	Maximum number of tries when the JSON received on GET is invalid.		10	✓	✓	✓	
invalid-json-on-get-sleep-initial-ms	Initial sleep in milliseconds between retries when the JSON received on GET is invalid.		10000	✓	✓	✓	
invalid-json-on-get-sleep-max-ms	Maximum sleep in milliseconds between retries when the JSON received on GET is invalid.		300000	✓	✓	✓	
invalid-json-on-get-sleep-multiplicator	Multiplication factor for sleep between retries when the JSON received on GET is invalid.		2	✓	✓	✓	
invalid-json-on-post-max-tries	Maximum number of tries when the JSON received on POST is invalid.		1	✓	✓	✓	
invalid-json-on-post-sleep-initial-ms	Initial sleep in milliseconds between retries when the JSON received on POST is invalid.		10000	✓	✓	✓	
invalid-json-on-post-sleep-max-ms	Maximum sleep in milliseconds between retries when the JSON received on POST is invalid.		300000	✓	✓	✓	
invalid-json-on-post-sleep-multiplicator	Multiplication factor for sleep between retries when the JSON received on POST is invalid.		2	✓	✓	✓	
invantive-sql-compress-sparse-arrays	Whether to compress sparse arrays in result sets during compression.	SQL Engine V1	True	✓	✓	✓	
invantive-sql-correct-invalid-date	Whether to correct dates considered invalid since they are before 01-01-1753. When nullable, they are removed. Otherwise they are replaced by 01-01-1753.	SQL Engine V1	False	✓	✓	✓	
invantive-sql-forward-filters-to-data-containers	Whether to forward filters to data containers.	SQL Engine V1	True	✓	✓	✓	
invantive-sql-share-byte-arrays	Whether to share the memory used by identical byte arrays in result sets during compression.	SQL Engine V1	True	✓	✓	✓	
invantive-sql-share-strings	Whether to share the memory used by identical strings in result sets during compression.	SQL Engine V1	True	✓	✓	✓	
invantive-sql-shuffle-fetch-results-data-containers	Whether to shuffle results fetched from data containers.	SQL Engine V1	False	✓	✓	✓	
invantive-use-cache	Whether to cache the results of a query.	SQL Engine V1	True	✓	✓	✓	
join-set-points-per-request	Maximum number of values in a request when executing a join set.	OData	60	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
limit-partition-calls-left	Minimum number of remaining API calls on a partition towards a hard limit. When below, an error is raised.	OData	500	✓	✓	✓	
log-native-calls-to-disk-max-events	Maximum number of events to register from last activation.	Shared		✓	✓	✓	
log-native-calls-to-disk-max-seconds	Maximum number of seconds to register from last activation.	Shared		✓	✓	✓	
log-native-calls-to-disk-on-error	Registers native calls to data container backend as disk files when an error occurred.	Shared	False	✓	✓	✓	
log-native-calls-to-disk-on-success	Registers native calls to data container backend as disk files when successful.	Shared	False	✓	✓	✓	
log-native-calls-to-trace	Log native calls to data container backend on the trace.	Shared	False	✓	✓	✓	
maximum-length-identifiers	Non-default maximum length in characters of identifier names.	Shared		✓	✓	✓	
max-odata-filters	The maximum number of OData filter elements.	OData	100	✓	✓	✓	
max-url-length-accepted	The maximum accepted URL length before raising an error.	Shared	8000	✓	✓	✓	
max-url-length-desired	The maximum desired URL length.	Shared	8000	✓	✓	✓	
metadata-cache-max-age-sec	Maximum acceptable age in seconds for re-use of metadata.	OData		✓	✓	✓	
oauth-unauthorized-max-tries	Maximum number of tries when an OAuth exception occurs.	OData	2	✓	✓	✓	
oauth-unauthorized-sleep-initial-ms	Initial sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails.	OData	10000	✓	✓	✓	
oauth-unauthorized-sleep-max-ms	Maximum sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails.	OData	1000	✓	✓	✓	
oauth-unauthorized-sleep-multiplicator	Multiplication factor for sleep between OAuth reauthentication tries when the OAuth authentication fails.	OData	2	✓	✓	✓	
partition-slot-based-rate-limit-length-ms	Total length in ms across all slots of a partition-based rate limit.	Shared	1100	✓		✓	
partition-slot-based-rate-limit-slots	Number of slots per partition-based rate limit. Null means no slot-based rate limit	Shared	1	✓		✓	
pre-request-delay-ms	Pre-request delay in milliseconds per request.	Shared	0	✓	✓	✓	
requested-page-size	Preferred number of rows to exchange per round trip; only	Shared		✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
	effective on limited platforms such as AFAS Online						
requests-parallel-max	Maximum number of parallel data requests from individual partitions on the data container.	Shared	32	✓	✓	✓	
simulate-http-400-errors	Simulate HTTP 400 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-400-errors-percentage	Percentage of simulated HTTP 400 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-401-errors	Simulate HTTP 401 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-401-errors-percentage	Percentage of simulated HTTP 401 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-403-errors	Simulate HTTP 403 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-403-errors-percentage	Percentage of simulated HTTP 403 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-408-errors	Simulate HTTP 408 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-408-errors-percentage	Percentage of simulated HTTP 408 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-429-errors	Simulate HTTP 429 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-429-errors-percentage	Percentage of simulated HTTP 429 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-500-errors	Simulate HTTP 500 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-500-errors-percentage	Percentage of simulated HTTP 500 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-502-errors	Simulate HTTP 502 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-502-errors-percentage	Percentage of simulated HTTP 502 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-503-errors	Simulate HTTP 503 errors when exchanging results with the OData endpoint.		False	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Driver's File	Set from Log On
simulate-http-503-errors-percentage	Percentage of simulated HTTP 503 errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-protocol-errors	Simulate HTTP protocol errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-protocol-errors-percentage	Percentage of simulated HTTP protocol errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
simulate-http-timeout-errors	Simulate HTTP timeout errors when exchanging results with the OData endpoint.		False	✓	✓	✓	
simulate-http-timeout-errors-percentage	Percentage of simulated HTTP timeout errors when exchanging results with the OData endpoint.		0	✓	✓	✓	
slot-based-rate-limit-length-ms	Total length in ms across all slots of a slot-based rate limit.	Shared	60000	✓		✓	
slot-based-rate-limit-slots	Number of slots of a slot-based rate limit. Null means no slot-based rate limit	Shared		✓		✓	
standardize-identifiers	Rewrite all identifiers to the preferred standards as configured by standardize-identifiers-casing and maximum-length-identifiers.	Shared	True	✓	✓	✓	
standardize-identifiers-casing	Rewrite all identifiers to the recommended standard platform-specific casing when changing a data model on a case-dependent platform.	Shared	True	✓	✓	✓	
use-batch-insert	Whether to use batch insert.	OData	True	✓	✓	✓	
use-http-disk-cache-read	Whether to use HTTP responses from previous queries stored on disk to answer the current query.	Shared	False	✓	✓	✓	
use-http-disk-cache-write	Whether to memorize HTTP responses on disk.	Shared	False	✓	✓	✓	
use-http-memory-cache-read	Whether to use HTTP responses from previous queries stored in memory that can answer the current query.	OData	True	✓	✓	✓	
use-http-memory-cache-write	Whether to memorize HTTP responses from previous queries for use by future queries.	OData	True	✓	✓	✓	
use-test-environment	Use the test environment. If false or null, the production environment will be used.	Roller	False	✓		✓	✓

## 3 Schema: Data

### 3.1 Tables

#### 3.1.1 AttendanceLocationsByDate: Roller Attendance Location by Date

Catalog: Roller

Schema: Data

Label: Attendance Location by Date

Documentation:

The timespan covered by the start and end date can not exceed 1 day.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/attendances`

Insert Roller API URL: `data/attendances`

Update Roller API URL: `data/attendances`

Delete Roller API URL: `data/attendances`

Field Selection Method: `NotRequired`

Base Path: `LocationIds[*]`

### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `AttendanceLocationsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>endDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
<code>startDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

### Columns of Table Function

The columns of the table function `AttendanceLocationsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingCustomerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
bookingItemId	string		<input checked="" type="checkbox"/>	Unique identifier for a ticket.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
checkInDateTime	datetime	Check-in	<input checked="" type="checkbox"/>	Date/time of the attendance.
customerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the specific ticket.
employeeId	string	Employee ID	<input type="checkbox"/>	The identifier of the staff member who completed the check in.
locationId	string		<input checked="" type="checkbox"/>	Location the ticket operates in the venue.
parentProductId	string	Parent Product ID	<input checked="" type="checkbox"/>	Unique identifier of the parent product being checked in.
productId	string	Product ID	<input checked="" type="checkbox"/>	Unique identifier of the product being checked in.
productName	string	Product Name	<input checked="" type="checkbox"/>	
receiptNumber	string	Receipt Number	<input checked="" type="checkbox"/>	
sessionEnd	string	Session End	<input type="checkbox"/>	End time of the ticket if applicable.
sessionStart	string	Session Start	<input type="checkbox"/>	Start time of the ticket if applicable.

### 3.1.2 AttendancesByDate: Roller Attendances by Date

Catalog: Roller

Schema: Data

Label: Attendances by Date

Documentation:

The timespan covered by the start and end date can not exceed 1 day.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/attendances`

Insert Roller API URL: `data/attendances`

Update Roller API URL: `data/attendances`

Delete Roller API URL: `data/attendances`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `AttendancesByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a

pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `AttendancesByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingCustomerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
bookingItemId	string		<input checked="" type="checkbox"/>	Unique identifier for a ticket.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
checkInDateTime	datetime	Check-in	<input checked="" type="checkbox"/>	Date/time of the attendance.
customerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the specific ticket.
employeeId	string	Employee ID	<input type="checkbox"/>	The identifier of the staff member who completed the check in.
parentProductId	string	Parent Product ID	<input checked="" type="checkbox"/>	Unique identifier of the parent product being checked in.
productId	string	Product ID	<input checked="" type="checkbox"/>	Unique identifier of the product being checked in.
productName	string	Product Name	<input checked="" type="checkbox"/>	
receiptNumber	string	Receipt Number	<input checked="" type="checkbox"/>	
sessionEnd	string	Session End	<input type="checkbox"/>	End time of the ticket if applicable.
sessionStart	string	Session Start	<input type="checkbox"/>	Start time of the ticket if applicable.

### 3.1.3 BookingById: Roller Booking by ID

Catalog: Roller

Schema: Data

Label: Booking by ID

Documentation:

Booking details.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `bookings/{uniqueIdOrBookingId}`

Insert Roller API URL: `bookings/{uniqueIdOrBookingId}`

Update Roller API URL: `bookings/{uniqueIdOrBookingId}`

Delete Roller API URL: `bookings/{uniqueIdOrBookingId}`

Field Selection Method: `NotRequired`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingById`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>uniqueIdOrBookingId</code>	string	<input type="checkbox"/>		ID of the booking to get.

## Columns of Table Function

The columns of the table function `BookingById` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>bookingReference</code>	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
<code>comments</code>	string		<input type="checkbox"/>	Additional information about the booking.
<code>companyId</code>	string		<input type="checkbox"/>	The Id of the company who made the booking.
<code>createdDate</code>	datetime	Created	<input checked="" type="checkbox"/>	Date the booking item was created.
<code>customerId</code>	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the booking.

Name	Data Type	Label	Required	Documentation
discount	decimal	Discount	<input type="checkbox"/>	Total discount applied.
externalld	string		<input type="checkbox"/>	External booking ID.
fees	decimal		<input type="checkbox"/>	Total fees applied.
name	string	Name	<input type="checkbox"/>	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
remainder	decimal		<input type="checkbox"/>	Total remaining cost/outstanding balance.
source	string		<input type="checkbox"/>	The application that created the booking.
status	string		<input type="checkbox"/>	Booking status.
total	decimal	Total	<input type="checkbox"/>	Total cost.
uniqueID	guid		<input type="checkbox"/>	Unique booking ID.

### 3.1.4 BookingItems: Roller Booking Items

Catalog: Roller

Schema: Data

Label: Booking Items

Documentation:

Search bookings by keywords, date and more. At least one search criteria is required. If there are more than 100 bookings only the latest 100 bookings are returned.

This is a read-only table function. The Roller API may not support changing the data or the Invantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `bookings`

Insert Roller API URL: `bookings`

Update Roller API URL: `bookings`

Delete Roller API URL: `bookings`

Field Selection Method: `NotRequired`

Base Path: `bookings[*].item[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingItems`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four

parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
date	datetime	<input type="checkbox"/>		Date of the bookings you are looking for.
keyw ords	string	<input type="checkbox"/>		Keyw ords match against various properties of a booking including customer detail. Up to 10 bookings are returned. The new est bookings are returned w hen there are more than 10 matches found.
locationIds	string	<input type="checkbox"/>		Comma separated capacity location IDs. Date is required.
productIds	string	<input type="checkbox"/>		Comma separated product IDs. Date is required.
startTime	string	<input type="checkbox"/>		Session start time (only applies to session products) e.g. 09:00 = 9am, 13:00 = 1pm. Date is required.

## Columns of Table Function

The columns of the table function `BookingItems` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
booking_bookingReference	string		<input checked="" type="checkbox"/>	Unique identifier for a booking.
booking_createdDate	datetime		<input checked="" type="checkbox"/>	Date the booking item w as created.
booking_customerId	string		<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
booking_name	string		<input type="checkbox"/>	Name of the booking can be different to the booking ow ner (defaults to the booking ow ner's name).
booking_status	string		<input type="checkbox"/>	Booking status.
booking_total	decimal		<input type="checkbox"/>	Total cost.
booking_uniqueID	guid		<input type="checkbox"/>	Unique booking ID.
bookingDate	datetime	Booking Date	<input type="checkbox"/>	The date this item w as purchased or the date this item is valid for.
productId	int32	Product ID	<input type="checkbox"/>	Unique identifier for a product.
quantity	int32	Quantity	<input type="checkbox"/>	Number of items purchased - same as number of tickets except w hen GroupSize is greater than one.

Name	Data Type	Label	Required	Documentation
startTime	string		<input type="checkbox"/>	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

### 3.1.5 BookingItemsByBookingId: Roller Booking Items by ID

Catalog: Roller

Schema: Data

Label: Booking Items by ID

Documentation:

Booking item details.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `bookings/{uniqueIdOrBookingId}`

Insert Roller API URL: `bookings/{uniqueIdOrBookingId}`

Update Roller API URL: `bookings/{uniqueIdOrBookingId}`

Delete Roller API URL: `bookings/{uniqueIdOrBookingId}`

Field Selection Method: NotRequired

Base Path: `items[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingItemsByBookingId`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
uniqueIdOrBookingId	string	<input type="checkbox"/>		ID of the booking to get.

## Columns of Table Function

The columns of the table function `BookingItemsByBookingId` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
booking_comments	string		<input type="checkbox"/>	Additional information about the booking.
booking_companyId	string		<input type="checkbox"/>	The Id of the company who made the booking.
booking_createdDate	datetime		<input checked="" type="checkbox"/>	Date the booking item was created.
booking_customerId	string		<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
booking_discount	decimal		<input type="checkbox"/>	Total discount applied.
booking_externalId	string		<input type="checkbox"/>	External booking ID.
booking_fees	decimal		<input type="checkbox"/>	Total fees applied.
booking_name	string		<input type="checkbox"/>	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
booking_remainder	decimal		<input type="checkbox"/>	Total remaining cost/outstanding balance.
booking_source	string		<input type="checkbox"/>	The application that created the booking.
booking_status	string		<input type="checkbox"/>	Booking status.
booking_total	decimal		<input type="checkbox"/>	Total cost.
booking_uniqueID	guid		<input type="checkbox"/>	Unique booking ID.
bookingDate	datetime	Booking Date	<input type="checkbox"/>	The date this item was purchased or the date this item is valid for.
bookingEndDate	datetime		<input type="checkbox"/>	The date this item is valid until (this can change after first check-in dependend in product configuration).
bookingItemId	int32		<input type="checkbox"/>	Unique identifier for an item.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
endTime	string		<input type="checkbox"/>	Session end time in 24hr format e.g. 13:30 = 1:30 PM.
productId	int32	Product ID	<input type="checkbox"/>	Unique identifier for a product.
quantity	int32	Quantity	<input type="checkbox"/>	Number of items purchased - same as number of tickets except when GroupSize is greater than one.
startTime	string		<input type="checkbox"/>	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

### 3.1.6 BookingItemsByDate: Roller Booking Items by Date

Catalog: Roller

Schema: Data

Label: Booking Items by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/bookingitems`

Insert Roller API URL: `data/bookingitems`

Update Roller API URL: `data/bookingitems`

Delete Roller API URL: `data/bookingitems`

Field Selection Method: `NotRequired`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingItemsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>endDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
<code>startDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `BookingItemsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>bookingCreatedByStaffId</code>	<code>string</code>		<input type="checkbox"/>	Staff Id that created the Booking.
<code>bookingCreatedDate</code>	<code>datetime</code>		<input checked="" type="checkbox"/>	Date the booking was created.
<code>bookingCustomerId</code>	<code>string</code>	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
<code>bookingDate</code>	<code>datetime</code>	Booking Date	<input checked="" type="checkbox"/>	Booking date of the ticket - when the ticket is valid for/from.

Name	Data Type	Label	Required	Documentation
bookingLocation	string	Location	<input checked="" type="checkbox"/>	Sales channel the booking was created through.
bookingModifiedDate	datetime	Modified	<input checked="" type="checkbox"/>	Date the booking was most recently modified.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
bookingStatus	string	Status	<input checked="" type="checkbox"/>	Payment status of the booking.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date the booking item was created.
discountAmount	decimal		<input type="checkbox"/>	The amount discounted against this specific booking item.
groupSize	int32	Group Size	<input checked="" type="checkbox"/>	The number of guests per quantity - eg. GroupSize of 4 with Quantity 3 would be 12 guests.
productId	string	Product ID	<input checked="" type="checkbox"/>	Unique identifier of the product.
quantity	int32	Quantity	<input checked="" type="checkbox"/>	Quantity of the product on this booking item.
sessionEnd	string	Session End	<input type="checkbox"/>	End time of the item if applicable.
sessionStart	string	Session Start	<input type="checkbox"/>	Start time of the item if applicable.
taxExemptId	string		<input type="checkbox"/>	The Tax ID provided when overriding the tax amount.

### 3.1.7 BookingPaymentsByDate: Roller Booking Payments by Date

Catalog: Roller

Schema: Data

Primary Keys: BookingPaymentId

Label: Booking Payments by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/bookingpayments`

Insert Roller API URL: `data/bookingpayments`

Update Roller API URL: `data/bookingpayments`

Delete Roller API URL: `data/bookingpayments`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingPaymentsByDate`. A value must be provided at all times for required parameters, but

optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `BookingPaymentsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
authorizingStaffId	string		<input type="checkbox"/>	StaffId of the staff member who provided their passcode to process a refund.
bookingPaymentId	string	Booking Payment ID	<input checked="" type="checkbox"/>	Unique Identifier of a transaction.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date of the transaction.
creditCardLast4Digits	string		<input type="checkbox"/>	Last 4 digits of the credit card used to take payment.
deviceId	string		<input type="checkbox"/>	Unique Identifier of the POS/SSK device used to process the transaction.
receiptNumber	string	Receipt Number	<input checked="" type="checkbox"/>	
staffId	string	Staff ID	<input type="checkbox"/>	Identifier of the staff member who processed the transaction.
tip	decimal	Tip	<input type="checkbox"/>	Gratuity amount against the transaction.
total	decimal	Total	<input checked="" type="checkbox"/>	Value of the transaction.
transactionFeeAmount	decimal	Transaction Fee Amount	<input type="checkbox"/>	The value of the transaction fee against a transaction.
transactionId	string	Transaction ID	<input type="checkbox"/>	The unique payment identifier from the payment gateway - or gift card number for gift card payments.

### 3.1.8 Bookings: Roller Bookings

Catalog: Roller

Schema: Data

Label: Bookings

Documentation:

Search bookings by keywords, date and more. At least one search criteria is required. If there are more than 100 bookings only the latest 100 bookings are returned.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `bookings`

Insert Roller API URL: `bookings`

Update Roller API URL: `bookings`

Delete Roller API URL: `bookings`

Field Selection Method: `NotRequired`

Base Path: `bookings[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `Bookings`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>date</code>	<code>datetime</code>	<input type="checkbox"/>		Date of the bookings you are looking for.
<code>keywords</code>	<code>string</code>	<input type="checkbox"/>		Keywords match against various properties of a booking including customer detail. Up to 10 bookings are returned. The newest bookings are returned when there are more than 10 matches found.
<code>locationids</code>	<code>string</code>	<input type="checkbox"/>		Comma separated capacity location IDs. Date is required.
<code>productids</code>	<code>string</code>	<input type="checkbox"/>		Comma separated product IDs. Date is required.

Name	Data Type	Required	Default Value	Documentation
startTime	string	<input type="checkbox"/>		Session start time (only applies to session products) e.g. 09:00 = 9am, 13:00 = 1pm. Date is required.

## Columns of Table Function

The columns of the table function `Bookings` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date the booking item was created.
customerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the booking.
name	string	Name	<input type="checkbox"/>	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
status	string		<input type="checkbox"/>	Booking status.
total	decimal	Total	<input type="checkbox"/>	Total cost.
uniqueID	guid		<input type="checkbox"/>	Unique booking ID.

### 3.1.9 BookingSignedWaiversByDate: Roller Booking Signed Waivers by Date

Catalog: Roller

Schema: Data

Primary Keys: SignedWaiverId

Label: Booking Signed Waivers by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/bookingsignedwaivers`

Insert Roller API URL: `data/bookingsignedwaivers`

Update Roller API URL: `data/bookingsignedwaivers`

Delete Roller API URL: `data/bookingsignedwaivers`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `BookingSignedWaiversByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `BookingSignedWaiversByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date the <code>BookingSignedWaiver</code> record was created.
modifiedDate	datetime	Modified	<input checked="" type="checkbox"/>	Most recent date the <code>BookingSignedWaiver</code> record was modified.
receiptNumber	string	Receipt Number	<input checked="" type="checkbox"/>	
signedWaiverId	string	Signed Waiver ID	<input checked="" type="checkbox"/>	Unique identifier for the waiver record.
ticketId	string	Ticket ID	<input type="checkbox"/>	Unique identifier for a ticket within a booking.

### 3.1.10 CustomerById: Roller Customer by ID

Catalog: Roller

Schema: Data

Label: Customer by ID

Documentation:

Customer details.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `customers/{customerId}`

Insert Roller API URL: `customers/{customerId}`

Update Roller API URL: `customers/{customerId}`

Delete Roller API URL: `customers/{customerId}`

Field Selection Method: `NotRequired`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `CustomerById`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>customerId</code>	string	<input checked="" type="checkbox"/>		Unique customer ID.

## Columns of Table Function

The columns of the table function `CustomerById` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>acceptMarketing</code>	boolean	Accept Marketing	<input checked="" type="checkbox"/>	Whether or not the customer has accepted.
<code>addressCity</code>	string		<input type="checkbox"/>	City of the customer.
<code>addressCountry</code>	string		<input type="checkbox"/>	Country of the customer.
<code>addressPostcode</code>	string		<input type="checkbox"/>	Postcode of the customer.
<code>addressState</code>	string		<input type="checkbox"/>	State of the customer.
<code>addressStreet</code>	string		<input type="checkbox"/>	Street of the customer.
<code>addressSuburb</code>	string		<input type="checkbox"/>	Suburb of the customer.
<code>dateOfBirth</code>	datetime	Date of Birth	<input type="checkbox"/>	Date of birth of the customer.
<code>email</code>	string	Email	<input type="checkbox"/>	Email address of the customer.
<code>firstName</code>	string	First Name	<input type="checkbox"/>	First name of the customer.

Name	Data Type	Label	Required	Documentation
lastName	string	Last Name	<input type="checkbox"/>	Last name of the customer.
phone	string		<input type="checkbox"/>	Phone number of the customer.

### 3.1.11 CustomersByDate: Roller Customers by Date

Catalog: Roller

Schema: Data

Primary Keys: CustomerId

Label: Customers by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/customers`

Insert Roller API URL: `data/customers`

Update Roller API URL: `data/customers`

Delete Roller API URL: `data/customers`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `CustomersByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `CustomersByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>acceptMarketing</code>	boolean	Accept Marketing	<input checked="" type="checkbox"/>	Whether or not the customer has accepted.
<code>contactNumber</code>	string	Contact Number	<input type="checkbox"/>	Contact phone number of the customer.
<code>country</code>	string	Country	<input type="checkbox"/>	Country of address of the customer.
<code>createdDate</code>	datetime	Created	<input checked="" type="checkbox"/>	Date the customer record was created.
<code>customerId</code>	string	Customer ID	<input checked="" type="checkbox"/>	Unique identifier of the customer.
<code>dateOfBirth</code>	datetime	Date of Birth	<input type="checkbox"/>	Date of birth of the customer.
<code>email</code>	string	Email	<input type="checkbox"/>	Email address of the customer.
<code>firstName</code>	string	First Name	<input type="checkbox"/>	First name of the customer.
<code>gender</code>	string	Gender	<input type="checkbox"/>	Gender of the customer.
<code>lastName</code>	string	Last Name	<input type="checkbox"/>	Last name of the customer.
<code>modifiedDate</code>	datetime	Modified	<input checked="" type="checkbox"/>	Date the customer record was most recently modified.
<code>postcode</code>	string	ZIP Code	<input type="checkbox"/>	Postcode of the customer.

### 3.1.12 Devices

Catalog: Roller

Schema: Data

Primary Keys: `DeviceId`

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/devices`

Insert Roller API URL: `data/devices`

Update Roller API URL: `data/devices`

Delete Roller API URL: `data/devices`

Field Selection Method: `NotRequired`

## Table Columns

The columns of the Table `Devices` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>createdDate</code>	datetime	Created	<input type="checkbox"/>	The date the device was created.

Name	Data Type	Label	Required	Documentation
deviceId	string		<input type="checkbox"/>	Unique identifier of the POS Device.
deviceName	string		<input type="checkbox"/>	Name of the device.
deviceStatus	string		<input type="checkbox"/>	Active = Currently usable device, Inactive = Deleted device.
deviceType	int32		<input type="checkbox"/>	0 = POS device, 1 = SSK device, 2 = Check In Scanner device.

### 3.1.13 DiscountCodes: Roller Discount Codes

Catalog: Roller

Schema: Data

Label: Discount Codes

This is a read-only table. The Roller API may not support changing the data or the Invantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/discounts`

Insert Roller API URL: `data/discounts`

Update Roller API URL: `data/discounts`

Delete Roller API URL: `data/discounts`

Field Selection Method: `NotRequired`

Base Path: `codes [ * ]`

## Table Columns

The columns of the Table `DiscountCodes` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
discount_amountOff	decimal		<input type="checkbox"/>	Dollar amount off the product cost.
discount_bookingDateRestrictionDateRangeDays	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionDateRangeEndDate	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionDateRangeStartDate	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionFromNumber	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.

Name	Data Type	Label	Required	Documentation
discount_bookingDateRestrictionFromType	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingRuleNumberOfUses	string		<input type="checkbox"/>	The usage limits for each code per booking
discount_bookingRuleType	string		<input type="checkbox"/>	1 = Across selected products (quantity purchased counts towards uses).
discount_codeGenerationMode	string		<input checked="" type="checkbox"/>	How discount codes are generated.
discount_discountId	string		<input checked="" type="checkbox"/>	Unique identifier of the discount.
discount_endDate	datetime		<input type="checkbox"/>	Last day the discount can be used.
discount_isSingleUseCode	boolean		<input checked="" type="checkbox"/>	When true, a code for this discount can only be used once.
discount_maxApplicableAmount	decimal		<input type="checkbox"/>	Maximum discount that can be applied to a booking using this discount.
discount_name	string		<input checked="" type="checkbox"/>	Discount name.
discount_percentOff	decimal		<input type="checkbox"/>	Percentage off the product cost.
discount_startDate	datetime		<input type="checkbox"/>	First day the discount can be used.
discount_usageLimitNumberOfUses	string		<input type="checkbox"/>	The usage limits for each code in total.
discount_usageLimitType	string		<input type="checkbox"/>	0 = Per code, 1 = Per user, 2 = For each selected product (regardless of quantity purchased), 3 = Across selected products (quantity purchased counts towards uses), 4 = Per code per day, 5 = Per code per week, 6 = Per code per month, 7 = Per code per year.
discountCode	string	Discount Code	<input checked="" type="checkbox"/>	Discount code.

### 3.1.14 DiscountProducts: Roller Discount Products

Catalog: Roller

Schema: Data

Label: Discount Products

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/discounts`

Insert Roller API URL: `data/discounts`

Update Roller API URL: `data/discounts`

Delete Roller API URL: `data/discounts`

Field Selection Method: `NotRequired`

Base Path: `productIds[*]`

## Table Columns

The columns of the Table `DiscountProducts` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>discount_amountOff</code>	decimal		<input type="checkbox"/>	Dollar amount off the product cost.
<code>discount_bookingDateRestrictionDateRangeDays</code>	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
<code>discount_bookingDateRestrictionDateRangeEndDate</code>	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
<code>discount_bookingDateRestrictionDateRangeStartDate</code>	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
<code>discount_bookingDateRestrictionFromNumber</code>	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
<code>discount_bookingDateRestrictionFromType</code>	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
<code>discount_bookingRuleNumberOfUses</code>	string		<input type="checkbox"/>	The usage limits for each code per booking
<code>discount_bookingRuleType</code>	string		<input type="checkbox"/>	1 = Across selected products (quantity purchased counts towards uses).
<code>discount_codeGenerationMode</code>	string		<input checked="" type="checkbox"/>	How discount codes are generated.
<code>discount_discountId</code>	string		<input checked="" type="checkbox"/>	Unique identifier of the discount.
<code>discount_endDate</code>	datetime		<input type="checkbox"/>	Last day the discount can be used.
<code>discount_isSingleUseCode</code>	boolean		<input checked="" type="checkbox"/>	When true, a code for this discount can only be used once.
<code>discount_maxApplicableAmount</code>	decimal		<input type="checkbox"/>	Maximum discount that can be applied to a booking using this discount.
<code>discount_name</code>	string		<input checked="" type="checkbox"/>	Discount name.
<code>discount_percentOff</code>	decimal		<input type="checkbox"/>	Percentage off the product cost.
<code>discount_startDate</code>	datetime		<input type="checkbox"/>	First day the discount can be used.
<code>discount_usageLimitNumberOfUses</code>	string		<input type="checkbox"/>	The usage limits for each code in total.

Name	Data Type	Label	Required	Documentation
discount_usageLimitType	string		<input type="checkbox"/>	0 = Per code, 1 = Per user, 2 = For each selected product (regardless of quantity purchased), 3 = Across selected products (quantity purchased counts towards uses), 4 = Per code per day, 5 = Per code per week, 6 = Per code per month, 7 = Per code per year.
productId	string	Product ID	<input checked="" type="checkbox"/>	Product ID.

### 3.1.15 Discounts: Roller Discounts

Catalog: Roller

Schema: Data

Primary Keys: DiscountId

Label: Discounts

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/discounts`

Insert Roller API URL: `data/discounts`

Update Roller API URL: `data/discounts`

Delete Roller API URL: `data/discounts`

Field Selection Method: NotRequired

## Table Columns

The columns of the Table `Discounts` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
amountOff	decimal	Discount Amount	<input type="checkbox"/>	Dollar amount off the product cost.
bookingDateRestrictionDateRangeDays	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionDateRangeEndDate	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionDateRangeStartDate	datetime		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionFromNumber	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date

Name	Data Type	Label	Required	Documentation
				the booking is for not the date the booking is created.
bookingDateRestrictionFromType	string		<input type="checkbox"/>	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingRuleNumberOfUses	string		<input type="checkbox"/>	The usage limits for each code per booking
bookingRuleType	string		<input type="checkbox"/>	1 = Across selected products (quantity purchased counts towards uses).
codeGenerationMode	string	Code Generation Mode	<input checked="" type="checkbox"/>	How discount codes are generated.
discountId	string	Discount ID	<input checked="" type="checkbox"/>	Unique identifier of the discount.
endDate	datetime	End Date	<input type="checkbox"/>	Last day the discount can be used.
isSingleUseCode	boolean	Is Single Use Code	<input checked="" type="checkbox"/>	When true, a code for this discount can only be used once.
maxApplicableAmount	decimal	Maximum Applicable Amount	<input type="checkbox"/>	Maximum discount that can be applied to a booking using this discount.
name	string	Name	<input checked="" type="checkbox"/>	Discount name.
percentOff	decimal	Percent Discount	<input type="checkbox"/>	Percentage off the product cost.
startDate	datetime	Start Date	<input type="checkbox"/>	First day the discount can be used.
usageLimitNumberOfUses	string		<input type="checkbox"/>	The usage limits for each code in total.
usageLimitType	string		<input type="checkbox"/>	0 = Per code, 1 = Per user, 2 = For each selected product (regardless of quantity purchased), 3 = Across selected products (quantity purchased counts towards uses), 4 = Per code per day, 5 = Per code per week, 6 = Per code per month, 7 = Per code per year.

### 3.1.16 GiftCardsByDate: Roller Gift Cards by Date

Catalog: Roller

Schema: Data

Primary Keys: GiftCardId

Label: Gift Cards by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/giftcards`

Insert Roller API URL: `data/giftcards`

Update Roller API URL: `data/giftcards`

Delete Roller API URL: `data/giftcards`

Field Selection Method: `NotRequired`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `GiftCardsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>endDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
<code>startDate</code>	<code>datetime</code>	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `GiftCardsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>balance</code>	<code>decimal</code>	Balance	<input checked="" type="checkbox"/>	The current balance of the gift card.
<code>bookingReference</code>	<code>string</code>	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
<code>createdDate</code>	<code>string</code>	Created	<input checked="" type="checkbox"/>	Date the gift card was created.
<code>expiryDate</code>	<code>string</code>	Expires	<input type="checkbox"/>	Date the gift card expires.
<code>giftCardId</code>	<code>string</code>	Gift Card ID	<input checked="" type="checkbox"/>	Unique identifier of the gift card.
<code>giftCardNumber</code>	<code>string</code>	Gift Card Number	<input checked="" type="checkbox"/>	The identifier used to redeem the gift card - unique per venue.
<code>initialValue</code>	<code>decimal</code>	Initial Value	<input checked="" type="checkbox"/>	The original balance of the gift card at purchase.
<code>lastUsedDate</code>	<code>string</code>	Last Used	<input type="checkbox"/>	Most recent date the gift card was redeemed.

Name	Data Type	Label	Required	Documentation
modifiedDate	string	Modified	<input checked="" type="checkbox"/>	Most recent date the gift card was modified.
ticketId	string	Ticket ID	<input checked="" type="checkbox"/>	Unique identifier for a ticket within a booking.

### 3.1.17 Locations: Roller Locations

Catalog: Roller

Schema: Data

Primary Keys: LocationId

Label: Locations

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/locations`

Insert Roller API URL: `data/locations`

Update Roller API URL: `data/locations`

Delete Roller API URL: `data/locations`

Field Selection Method: NotRequired

## Table Columns

The columns of the Table `Locations` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
allow MultipleBookings	boolean	Allow Multiple Bookings	<input checked="" type="checkbox"/>	Whether the location allows multiple different bookings to occur within it at the same time.
locationId	string		<input checked="" type="checkbox"/>	Unique identifier of the location.
name	string	Name	<input type="checkbox"/>	Location name.
parentLocationId	string	Parent Location ID	<input checked="" type="checkbox"/>	Unique identifier of the location group if applicable.

### 3.1.18 MembershipCreditsByDate: Roller Membership Statuses

Catalog: Roller

Schema: Data

Label: Membership Statuses

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/membershipcredits`

Insert Roller API URL: `data/membershipcredits`

Update Roller API URL: `data/membershipcredits`

Delete Roller API URL: `data/membershipcredits`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `MembershipCreditsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>bookingReferences</code>	string	<input type="checkbox"/>		Unique identifier for a booking
<code>date</code>	datetime	<input type="checkbox"/>		Search date, based on membership redemption date.

## Columns of Table Function

The columns of the table function `MembershipCreditsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>bookingReference</code>	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
<code>comment</code>	string		<input type="checkbox"/>	The note provided by the staff member issuing the credit.
<code>createdDate</code>	datetime	Created	<input checked="" type="checkbox"/>	Date the waiver record was created.
<code>creditValue</code>	decimal		<input checked="" type="checkbox"/>	Value of the credit issued.
<code>modifiedDate</code>	datetime	Modified	<input checked="" type="checkbox"/>	Date the waiver record was most recently modified.
<code>status</code>	string		<input type="checkbox"/>	The status of the membership credit - can be 'new', 'applied' or 'voided'.
<code>ticketId</code>	string	Ticket ID	<input type="checkbox"/>	Unique identifier for a ticket.

### 3.1.19 MembershipRedemptionsByDate: Roller Membership Redemptions

Catalog: Roller

Schema: Data

Label: Membership Redemptions

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/membershipredemptions`

Insert Roller API URL: `data/membershipredemptions`

Update Roller API URL: `data/membershipredemptions`

Delete Roller API URL: `data/membershipredemptions`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `MembershipRedemptionsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
<code>bookingReferences</code>	string	<input type="checkbox"/>		Unique identifier for a booking
<code>date</code>	datetime	<input type="checkbox"/>		Search date, based on membership redemption date.

## Columns of Table Function

The columns of the table function `MembershipRedemptionsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>bookingReference</code>	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
<code>redemptionDate</code>	datetime		<input checked="" type="checkbox"/>	The date/time of the redemption.
<code>ticketId</code>	string	Ticket ID	<input type="checkbox"/>	Unique identifier for a ticket.

### 3.1.20 MembershipStatusesByDate: Roller Membership Statuses

Catalog: Roller

Schema: Data

Label: Membership Statuses

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/membershipstatuses`

Insert Roller API URL: `data/membershipstatuses`

Update Roller API URL: `data/membershipstatuses`

Delete Roller API URL: `data/membershipstatuses`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `MembershipStatusesByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
bookingReferences	string	<input type="checkbox"/>		Unique identifier for a booking
date	datetime	<input type="checkbox"/>		Search date, based on membership redemption date.

## Columns of Table Function

The columns of the table function `MembershipStatusesByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
customTicketId	string		<input type="checkbox"/>	Unique identifier for a custom ticket.
eventDate	datetime	Event Date	<input checked="" type="checkbox"/>	The date/time of the membership event.

Name	Data Type	Label	Required	Documentation
nextStatus	string		<input type="checkbox"/>	The membership status on the membership after.
previousStatus	string		<input type="checkbox"/>	The membership status on the membership before.
ticketId	string	Ticket ID	<input type="checkbox"/>	Unique identifier for a ticket.

### 3.1.21 Modifiers

Catalog: Roller

Schema: Data

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/modifiers`

Insert Roller API URL: `data/modifiers`

Update Roller API URL: `data/modifiers`

Delete Roller API URL: `data/modifiers`

Field Selection Method: `NotRequired`

## Table Columns

The columns of the Table `Modifiers` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
amount	decimal		<input type="checkbox"/>	The cost of the modifier (as either a \$ amount or %).
modifierGroupId	string		<input type="checkbox"/>	Unique identifier of the modifier group
modifierGroupName	string		<input type="checkbox"/>	Name of the modifier group.
modifierId	string		<input type="checkbox"/>	Unique identifier of the modifier
modifierName	string		<input type="checkbox"/>	Name of the modifier.
modifierType	string		<input type="checkbox"/>	Whether the modifier adds a specific \$ amount of a % of the cost of the modified item.

### 3.1.22 ProductAvailabilitiesByDate: Roller Product Availabilities by Date

Catalog: Roller

Schema: Data

Label: Product Availabilities by Date

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: `NotRequired`

Base Path: `availabilities[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilitiesByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function `ProductAvailabilitiesByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
date	datetime		<input checked="" type="checkbox"/>	Availability date.
onlineSalesOpen	boolean		<input checked="" type="checkbox"/>	Indicator of whether the product is within its sale period.
productavailability_description	string(150)		<input type="checkbox"/>	Brief description of product.
productavailability_id	string		<input checked="" type="checkbox"/>	Product Id.
productavailability_imageUrl	string		<input type="checkbox"/>	Product image.
productavailability_name	string		<input checked="" type="checkbox"/>	Name.
productavailability_type	string		<input checked="" type="checkbox"/>	Product type.

### 3.1.23 ProductAvailabilityAllocationsByDate: Roller Product Availability Product Session Allocations by Date

Catalog: Roller

Schema: Data

Label: Product Availability Product Session Allocations by Date

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: `NotRequired`

Base Path: `sessions[*].allocations[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilityAllocationsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function `ProductAvailabilityAllocationsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
capacityRemaining	int32		<input type="checkbox"/>	Capacity remaining. Null value means unlimited capacity. For single-booking location capacity

Name	Data Type	Label	Required	Documentation
				the capacity is the number of locations available e.g. if you have 2 party rooms that hold 20 people each, CapacityRemaining = 2.
date	datetime		<input checked="" type="checkbox"/>	Availability date.
onlineSalesOpen	boolean		<input checked="" type="checkbox"/>	Indicator of whether the product is within its sale period.
productavailability_description	string(150)		<input type="checkbox"/>	Brief description of product.
productavailability_id	string		<input checked="" type="checkbox"/>	Product Id.
productavailability_imageUrl	string		<input type="checkbox"/>	Product image.
productavailability_name	string		<input checked="" type="checkbox"/>	Name.
productavailability_type	string		<input checked="" type="checkbox"/>	Product type.
productid	string	Product ID	<input checked="" type="checkbox"/>	Product Id.

### 3.1.24 ProductAvailabilityByDate: Roller Product Availability by Date

Catalog: Roller

Schema: Data

Label: Product Availability by Date

This is a read-only table function. The Roller API may not support changing the data or the Invariant SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilityByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function `ProductAvailabilityByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
description	string(150)		<input type="checkbox"/>	Brief description of product.
id	string		<input checked="" type="checkbox"/>	Product Id.
imageUrl	string		<input type="checkbox"/>	Product image.
name	string	Name	<input checked="" type="checkbox"/>	Name.
type	string		<input checked="" type="checkbox"/>	Product type.

### 3.1.25 ProductAvailabilityProductsByDate

Catalog: Roller

Schema: Data

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: `NotRequired`

Base Path: `products[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilityProductsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with `select \* from table(name1 => value1, name3 => value3)` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function `ProductAvailabilityProductsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
cost	decimal		<input type="checkbox"/>	Product cost.
description	string(150)		<input type="checkbox"/>	Brief description of product.
id	string		<input checked="" type="checkbox"/>	Product Id.
imageUrl	string		<input type="checkbox"/>	Product image.
name	string	Name	<input type="checkbox"/>	Name.
productavailability_description	string(150)		<input type="checkbox"/>	Brief description of product.
productavailability_id	string		<input checked="" type="checkbox"/>	Product Id.
productavailability_imageUrl	string		<input type="checkbox"/>	Product image.
productavailability_name	string		<input checked="" type="checkbox"/>	Name.
productavailability_type	string		<input checked="" type="checkbox"/>	Product type.
type	string		<input type="checkbox"/>	Product type.

### 3.1.26 ProductAvailabilitySessionAllocationsByDate: Roller Product Availability Product Session Allocations by Date

Catalog: Roller

Schema: Data

Label: Product Availability Product Session Allocations by Date

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: `NotRequired`

Base Path: `sessions[*].allocations[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilitySessionAllocationsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function

`ProductAvailabilitySessionAllocationsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
capacityRemaining	int32		<input type="checkbox"/>	Capacity remaining. Null value means unlimited capacity. For single-booking location capacity the capacity is the number of locations available e.g. if you have 2 party rooms that hold 20 people each, CapacityRemaining = 2.
productavailability_description	string(150)		<input type="checkbox"/>	Brief description of product.
productavailability_id	string		<input checked="" type="checkbox"/>	Product Id.
productavailability_imageUrl	string		<input type="checkbox"/>	Product image.
productavailability_name	string		<input checked="" type="checkbox"/>	Name.
productavailability_type	string		<input checked="" type="checkbox"/>	Product type.
productId	string	Product ID	<input checked="" type="checkbox"/>	Product Id.
session_capacityRemaining	int32		<input type="checkbox"/>	Summary of remaining capacity. Null value means unlimited capacity Check Allocations for individual remaining capacity per product (ticket type).

Name	Data Type	Label	Required	Documentation
session_date	datetime		<input checked="" type="checkbox"/>	Session date.
session_endTime	string		<input type="checkbox"/>	Session end time in 24hr format e.g. 11:30 = 11:30 AM.
session_name	string		<input type="checkbox"/>	Session name.
session_onlineSalesOpen	boolean		<input checked="" type="checkbox"/>	Indicator of whether the product is within its sale period.
session_startTime	string		<input type="checkbox"/>	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

### 3.1.27 ProductAvailabilitySessionsByDate: Roller Product Availability Product Sessions by Date

Catalog: Roller

Schema: Data

Label: Product Availability Product Sessions by Date

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `product-availability`

Insert Roller API URL: `product-availability`

Update Roller API URL: `product-availability`

Delete Roller API URL: `product-availability`

Field Selection Method: `NotRequired`

Base Path: `sessions[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `ProductAvailabilitySessionsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
Date	datetime	<input checked="" type="checkbox"/>		The date you need availability for.
ProductCategory	string	<input type="checkbox"/>		Finds product/availability for a given product category.

Name	Data Type	Required	Default Value	Documentation
ProductIds	string	<input type="checkbox"/>		Comma separated product IDs.

## Columns of Table Function

The columns of the table function `ProductAvailabilitySessionsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
capacityRemaining	int32		<input type="checkbox"/>	Summary of remaining capacity. Null value means unlimited capacity Check Allocations for individual remaining capacity per product (ticket type).
date	datetime		<input checked="" type="checkbox"/>	Session date.
endTime	string		<input type="checkbox"/>	Session end time in 24hr format e.g. 11:30 = 11:30 AM.
name	string	Name	<input type="checkbox"/>	Session name.
onlineSalesOpen	boolean		<input checked="" type="checkbox"/>	Indicator of whether the product is within its sale period.
productavailability_description	string(150)		<input type="checkbox"/>	Brief description of product.
productavailability_id	string		<input checked="" type="checkbox"/>	Product Id.
productavailability_imageUrl	string		<input type="checkbox"/>	Product image.
productavailability_name	string		<input checked="" type="checkbox"/>	Name.
productavailability_type	string		<input checked="" type="checkbox"/>	Product type.
startTime	string		<input type="checkbox"/>	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

### 3.1.28 Products: Roller Products

Catalog: Roller

Schema: Data

Primary Keys: ProductId

Label: Products

This is a read-only table. The Roller API may not support changing the data or the Invantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/products`

Insert Roller API URL: `data/products`

Update Roller API URL: `data/products`

Delete Roller API URL: `data/products`

Field Selection Method: NotRequired

## Table Columns

The columns of the Table `Products` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
cost	decimal		<input type="checkbox"/>	Product cost.
hqCode	string		<input type="checkbox"/>	Unique identifier of the HQ product it is linked to.
hqProductId	string		<input type="checkbox"/>	Unique identifier of the HQ parent product it is linked to.
name	string	Name	<input type="checkbox"/>	Product name.
parentProductId	string	Parent Product ID	<input type="checkbox"/>	Unique identifier of the parent product.
productId	string	Product ID	<input checked="" type="checkbox"/>	Unique identifier of the product.
productStatus	string		<input checked="" type="checkbox"/>	Published status of the product.
productSubType	string	Product Sub-type	<input type="checkbox"/>	The sub type/variation of the product.
productType	string	Product Type	<input checked="" type="checkbox"/>	The type/variation of the product.
reportingCategoryName	string	Reporting Category Name	<input type="checkbox"/>	Reporting category name the product is allocated to.

### 3.1.29 ReportingCategories: Roller Reporting Categories

Catalog: Roller

Schema: Data

Label: Reporting Categories

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/reportingcategories`

Insert Roller API URL: `data/reportingcategories`

Update Roller API URL: `data/reportingcategories`

Delete Roller API URL: `data/reportingcategories`

Field Selection Method: NotRequired

## Table Columns

The columns of the Table `ReportingCategories` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
glCode	string	General Ledger Account Code	<input type="checkbox"/>	General ledger (GL) code.
name	string	Name	<input checked="" type="checkbox"/>	Reporting Category name.

### 3.1.30 ReportingCategoryCategories: Roller Reporting Categories

Catalog: Roller

Schema: Data

Label: Reporting Categories

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/reportingcategories`

Insert Roller API URL: `data/reportingcategories`

Update Roller API URL: `data/reportingcategories`

Delete Roller API URL: `data/reportingcategories`

Field Selection Method: NotRequired

Base Path: `categories[*]`

## Table Columns

The columns of the Table `ReportingCategoryCategories` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>glCode</code>	string	General Ledger Account Code	<input type="checkbox"/>	General ledger (GL) code.
<code>name</code>	string	Name	<input checked="" type="checkbox"/>	Reporting Category name.
<code>parent_glCode</code>	string		<input type="checkbox"/>	Parent general ledger (GL) code.
<code>parent_name</code>	string		<input checked="" type="checkbox"/>	Parent reporting Category name.

### 3.1.31 ReportingCategoryProducts: Roller Reporting Products

Catalog: Roller

Schema: Data

Label: Reporting Products

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/reportingcategories`

Insert Roller API URL: `data/reportingcategories`

Update Roller API URL: `data/reportingcategories`

Delete Roller API URL: `data/reportingcategories`

Field Selection Method: NotRequired

Base Path: `products[*]`

## Table Columns

The columns of the Table `ReportingCategoryProducts` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>category_glCode</code>	string		<input type="checkbox"/>	General ledger (GL) code.
<code>category_name</code>	string		<input checked="" type="checkbox"/>	Reporting Category name.
<code>productId</code>	string	Product ID	<input checked="" type="checkbox"/>	Product ID.

### 3.1.32 Revenues: Roller Revenues

Catalog: Roller

Schema: Data

Primary Keys: `BookingPaymentId`

Label: Revenues

Documentation:

Revenues are filtered on event date instead on modified date.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/revenues`

Insert Roller API URL: `data/revenues`

Update Roller API URL: `data/revenues`

Delete Roller API URL: `data/revenues`

Field Selection Method: `NotRequired`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `Revenues`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
bookingReferences	string	<input type="checkbox"/>		Booking references specified as list in comma-separated format. Maximum limit is 100.
endDate	datetime	<input type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `Revenues` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
accountsReceivable	decimal	Account Receivable	<input checked="" type="checkbox"/>	The accounts receivable generated or removed for this entry.
bookingPaymentId	string	Booking Payment ID	<input type="checkbox"/>	Unique Identifier of a transaction.
bookingReference	string	Booking Reference	<input checked="" type="checkbox"/>	Unique identifier for a booking.
customerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer for the booking.
deferredRevenue	decimal	Deferred Revenue	<input type="checkbox"/>	The deferred revenue generated or removed for this entry.
deferredRevenueGiftCards	decimal		<input type="checkbox"/>	The giftcard deferred revenue generated or removed for this entry.
deferredRevenueOther	decimal		<input type="checkbox"/>	The external deferred revenue generated or removed for this entry (used for integrations, eg. Groupon).
discount	decimal	Discount	<input type="checkbox"/>	The value of the discount against a transaction entry.
discountCode	string	Discount Code	<input type="checkbox"/>	The discount code used.
eventDate	datetime	Event Date	<input checked="" type="checkbox"/>	The date the entry occurred - not necessarily the date the entry was generated, but when the values should be recorded as occurring.
eventType	string	Event Type	<input checked="" type="checkbox"/>	The type of entry - Transaction, Redemption or Expiry.
externalPaymentReference	string	External Payment Reference	<input type="checkbox"/>	The unique payment identifier from the payment gateway - or gift card number for gift card payments.
feeRevenue	decimal	Fee Revenue	<input checked="" type="checkbox"/>	The amount of revenue from transaction fees for this entry.
fundsReceived	decimal	Funds Received	<input checked="" type="checkbox"/>	The amount of money received for a transaction entry.
netRevenue	decimal	Net Revenue	<input checked="" type="checkbox"/>	The after tax revenue generated for this entry.

Name	Data Type	Label	Required	Documentation
paymentType	string	Payment Type	<input checked="" type="checkbox"/>	The type of tender used for the transaction.
productId	string	Product ID	<input checked="" type="checkbox"/>	Unique identifier of a product.
productType	string	Product Type	<input checked="" type="checkbox"/>	Passes, Add-ons, Custom Products, Gift Cards, Package, Wallet.
receiptNumber	string	Receipt Number	<input checked="" type="checkbox"/>	
recognisedDiscount	decimal	Recognized Discount	<input type="checkbox"/>	The value of the discount against a redemption or expiry entry.
taxOnFundsReceived	decimal	Tax on Funds Received	<input type="checkbox"/>	The amount of tax on funds received for a transaction entry.
taxPayable	decimal	Tax Payable	<input checked="" type="checkbox"/>	The amount of tax payable on recognised revenue for this entry.
taxPercent	decimal	Tax Percentage	<input checked="" type="checkbox"/>	The tax percent against the product in this entry.
ticketId	string	Ticket ID	<input checked="" type="checkbox"/>	Unique identifier for a ticket within a booking.
ticketTransactionValue	decimal	Ticket Transaction Value	<input checked="" type="checkbox"/>	The value of the amount paid against the booking that is being attributed to this entry.
ticketUnitCost	decimal	Ticket Unit Cost	<input checked="" type="checkbox"/>	The price of the ticket being referenced in the entry.
transactionDate	datetime	Transaction Date	<input checked="" type="checkbox"/>	Date of the first transaction against a booking.
transactionFeeAmount	decimal	Transaction Fee Amount	<input type="checkbox"/>	The value of the transaction fee against a transaction entry.
transactionLocation	string	Transaction Location	<input checked="" type="checkbox"/>	The sales channel the booking was generated from.

### 3.1.33 SignedWaiversByDate: Roller Signed Waivers by Date

Catalog: Roller

Schema: Data

Primary Keys: SignedWaiverId

Label: Signed Waivers by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/signedwaivers`

Insert Roller API URL: `data/signedwaivers`

Update Roller API URL: `data/signedwaivers`

Delete Roller API URL: `data/signedwaivers`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `SignedWaiversByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `SignedWaiversByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
contactNumber	string	Contact Number	<input type="checkbox"/>	Contact phone number of the waiver holder. Not returned for minors.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date the waiver record was created.
customerId	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer record of the waiver holder.
dateOfBirth	datetime	Date of Birth	<input type="checkbox"/>	Date of birth of the waiver holder.
email	string	Email	<input type="checkbox"/>	Email address of the waiver holder. Not returned for minors.
expiryDate	datetime	Expires	<input checked="" type="checkbox"/>	Date the waiver expired.
firstName	string	First Name	<input type="checkbox"/>	First name of the customer as completed on the waiver.
isForMinor	boolean	Is for Minor	<input checked="" type="checkbox"/>	Identifies whether the customer is a minor or not at the time of signing. Returns true for minors, false for adults.

Name	Data Type	Label	Required	Documentation
isValid	boolean	Is Valid	<input type="checkbox"/>	Only returned if w aiver verification is required. Returns true for verified w aivers and false for unverified.
lastName	string	Last Name	<input type="checkbox"/>	Second name of the customer as completed on the w aiver.
modifiedDate	datetime	Modified	<input checked="" type="checkbox"/>	Date the w aiver record w as most recently modified.
parentSignedWaiverId	string	Parent Signed Waiver ID	<input type="checkbox"/>	Only returned for minor's w aivers - w ill be the SignedWaiverId of their parent/guardians w aiver.
signedWaiverId	string	Signed Waiver ID	<input checked="" type="checkbox"/>	Unique identifier for the w aiver record.
w aiverId	string		<input checked="" type="checkbox"/>	The identifier of the version of the w aiver they have signed.

### 3.1.34 StaffMembers: Roller Staff Members

Catalog: Roller

Schema: Data

Primary Keys: staffId

Label: Staff Members

This is a read-only table. The Roller API may not support changing the data or the Invantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/staffs`

Insert Roller API URL: `data/staffs`

Update Roller API URL: `data/staffs`

Delete Roller API URL: `data/staffs`

Field Selection Method: NotRequired

## Table Columns

The columns of the Table `StaffMembers` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
contactNumber	string	Contact Number	<input type="checkbox"/>	Contact phone number of the staff member.
createdDate	datetime	Created	<input type="checkbox"/>	Date the staff record w as created.
email	string	Email	<input type="checkbox"/>	Email address of the staff member.
firstName	string	First Name	<input type="checkbox"/>	First name of the staff member.
lastName	string	Last Name	<input type="checkbox"/>	Last name of the staff member.

Name	Data Type	Label	Required	Documentation
role	string		<input type="checkbox"/>	Staff role / permission level.
staffId	string	Staff ID	<input checked="" type="checkbox"/>	Unique identifier for a staff member.

### 3.1.35 TicketDiscountsByDate: Roller Ticket Discounts by Date

Catalog: Roller

Schema: Data

Label: Ticket Discounts by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/tickets`

Insert Roller API URL: `data/tickets`

Update Roller API URL: `data/tickets`

Delete Roller API URL: `data/tickets`

Field Selection Method: NotRequired

Base Path: `DiscountIds[*]`

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `TicketDiscountsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `TicketDiscountsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>discountId</code>	string	Discount ID	<input checked="" type="checkbox"/>	Unique identifier of the discount valid for this ticket.
<code>ticket_bookingDate</code>	datetime		<input type="checkbox"/>	Booking date of the ticket - when the ticket is valid for/from.
<code>ticket_createdDate</code>	datetime		<input checked="" type="checkbox"/>	Date the ticket was created.
<code>ticket_customerId</code>	string		<input type="checkbox"/>	Unique identifier of the customer linked to the ticket.
<code>ticket_customTicketId</code>	string		<input type="checkbox"/>	Ticket identifier specified manually by the user e.g. the code from a pre-printed membership card.
<code>ticket_expiryDate</code>	datetime		<input checked="" type="checkbox"/>	Date the ticket is no longer valid.
<code>ticket_name</code>	string		<input type="checkbox"/>	Ticket holder name.
<code>ticket_numberOfRecurringPayments</code>	int32		<input type="checkbox"/>	Number of recurring payments.
<code>ticket_productId</code>	string		<input type="checkbox"/>	Unique identifier of the product the ticket is for.
<code>ticket_productSubType</code>	string		<input type="checkbox"/>	The sub type/variation of the product.
<code>ticket_productType</code>	string		<input checked="" type="checkbox"/>	The type/variation of the product.
<code>ticket_recurringPaymentFrequency</code>	string		<input type="checkbox"/>	Recurring payment frequency.
<code>ticket_ticketId</code>	string		<input checked="" type="checkbox"/>	Unique identifier of the ticket.

### 3.1.36 TicketsByDate: Roller Tickets by Date

Catalog: Roller

Schema: Data

Primary Keys: `TicketId`

Label: Tickets by Date

Documentation:

The timespan covered by the start and end date can not exceed 7 days.

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/tickets`

Insert Roller API URL: `data/tickets`

Update Roller API URL: `data/tickets`

Delete Roller API URL: `data/tickets`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `TicketsByDate`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Search end date, based on record modified date.
startDate	datetime	<input checked="" type="checkbox"/>		Search start date, based on record modified date.

## Columns of Table Function

The columns of the table function `TicketsByDate` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
bookingDate	datetime	Booking Date	<input type="checkbox"/>	Booking date of the ticket - w hen the ticket is valid for/from.
createdDate	datetime	Created	<input checked="" type="checkbox"/>	Date the ticket w as created.
customerld	string	Customer ID	<input type="checkbox"/>	Unique identifier of the customer linked to the ticket.
customTicketld	string		<input type="checkbox"/>	Ticket identifier specified manually by the user e.g. the code from a pre-printed membership card.
expiryDate	datetime	Expires	<input checked="" type="checkbox"/>	Date the ticket is no longer valid.
name	string	Name	<input type="checkbox"/>	Ticket holder name.
numberOfRecurringPayments	int32	Number of Recurring Payments	<input type="checkbox"/>	Number of recurring payments.
productld	string	Product ID	<input type="checkbox"/>	Unique identifier of the product the ticket is for.
productSubType	string	Product Sub-type	<input type="checkbox"/>	The sub type/variation of the product.
productType	string	Product Type	<input checked="" type="checkbox"/>	The type/variation of the product.
recurringPaymentFrequency	string	Recurring Payment Frequency	<input type="checkbox"/>	Recurring payment frequency.

Name	Data Type	Label	Required	Documentation
ticketId	string	Ticket ID	<input checked="" type="checkbox"/>	Unique identifier of the ticket.

### 3.1.37 TillReconciliations

Catalog: Roller

Schema: Data

This is a read-only table function. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/tillreconciliations`

Insert Roller API URL: `data/tillreconciliations`

Update Roller API URL: `data/tillreconciliations`

Delete Roller API URL: `data/tillreconciliations`

Field Selection Method: NotRequired

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function `TillReconciliations`. A value must be provided at all times for required parameters, but optional parameters in general do not need to have a value and the execution will default to a pre-defined behaviour. Values can be specified by position and by name. In both cases, all parameters not specified will be evaluated using their default values.

Value specification by position is done by listing all values from the first to the last needed value. For example: a ``select * from table(value1, value2, value3)`` on a table with four parameters will use the default value for the fourth parameter and the specified values for the first three.

Value specification by name is done by listing all values that require a value. For example with ``select * from table(name1 => value1, name3 => value3)`` on the same table will use the default values for the second and fourth parameters and the specified values for the first and third.

Name	Data Type	Required	Default Value	Documentation
endDate	datetime	<input checked="" type="checkbox"/>		Till closed date.

## Columns of Table Function

The columns of the table function `TillReconciliations` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
actualCashBalance	decimal		<input type="checkbox"/>	The amount of cash that was counted and reconciled as being in the till at the end of the till session.

Name	Data Type	Label	Required	Documentation
cardRefunds	decimal		<input type="checkbox"/>	Credit Card refunds for this till session.
cardTakings	decimal		<input type="checkbox"/>	Credit Card takings for this till session.
cashAdded	decimal		<input type="checkbox"/>	The amount of cash that was manually added to the till during the till session.
cashFloat	decimal		<input type="checkbox"/>	The amount of cash that was in the till at the beginning of the till session.
cashRefunds	decimal		<input type="checkbox"/>	Cash Refunds for this till session.
cashRemoved	decimal		<input type="checkbox"/>	The amount of cash that was manually removed from the till during the till session.
cashTakings	decimal		<input type="checkbox"/>	Cash takings for this till session.
cashVariance	decimal		<input type="checkbox"/>	The difference between ExpectedCashBalance and ActualCashBalance. A positive number indicates excess cash a negative number indicates a shortage.
checkTakings	decimal		<input type="checkbox"/>	Check takings for this till session.
deviceId	string		<input type="checkbox"/>	Unique identifier of the POS Device.
endDateDate	datetime		<input type="checkbox"/>	Date/time the till session was closed and reconciled.
expectedCashBalance	decimal		<input type="checkbox"/>	The amount of cash expected to be in the till at the end of the till session. This is CashFloat + CashAddedIn + CashTakings - Cash Refunds - CashTakenOut.
giftCardTakings	decimal		<input type="checkbox"/>	Gift Card takings for this till session (transactions paid for by gift cards).
startDate	datetime	Start Date	<input type="checkbox"/>	Date/time of the first transaction for this till session.

### 3.1.38 Venues

Catalog: Roller

Schema: Data

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `venues/me`

Insert Roller API URL: `venues/me`

Update Roller API URL: `venues/me`

Delete Roller API URL: `venues/me`

Field Selection Method: `NotRequired`

## Table Columns

The columns of the Table `Venues` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>codeOfConduct</code>	string		<input type="checkbox"/>	The terms and conditions set by the venue for guests when they make a purchase.
<code>country</code>	string	Country	<input type="checkbox"/>	Country where the venue is located.
<code>countryCode</code>	string		<input type="checkbox"/>	Country code where the venue is located.
<code>creditCardFeePercent</code>	decimal		<input type="checkbox"/>	The % amount of fees charged on Card payments on the Online Checkout and in Venue Manager.
<code>currency</code>	string		<input type="checkbox"/>	Trading currency.
<code>feePerTransaction</code>	decimal		<input type="checkbox"/>	The \$ amount of fees charged to guests for Online Checkout payments.
<code>fees</code>	decimal		<input type="checkbox"/>	Amount of booking fees applied to the booking.
<code>id</code>	int32		<input type="checkbox"/>	Roller Venue ID.
<code>locale</code>	string		<input type="checkbox"/>	Locale code where the venue is located.
<code>name</code>	string	Name	<input type="checkbox"/>	Name of the venue.
<code>paymentSettingsApiUrl</code>	string		<input type="checkbox"/>	URL for the payment service.
<code>paymentSettingsConfigurationId</code>	string		<input type="checkbox"/>	
<code>paymentSettingsIntegrationId</code>	string		<input type="checkbox"/>	
<code>productPricesIncludeTax</code>	boolean		<input type="checkbox"/>	Whether or not the prices of products for the venue include taxes.
<code>timeZone</code>	string		<input type="checkbox"/>	Local time zone.
<code>transactionFeePercent</code>	decimal		<input type="checkbox"/>	The % amount of fees charged to guests for Online Checkout payments.

### 3.1.39 Waivers: Roller Waivers

Catalog: `Roller`

Schema: `Data`

Primary Keys: `WaiverId`

Label: `Waivers`

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `data/waivers`

Insert Roller API URL: `data/waivers`

Update Roller API URL: `data/waivers`

Delete Roller API URL: `data/waivers`

Field Selection Method: `NotRequired`

## Table Columns

The columns of the Table `waivers` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>createdDate</code>	<code>datetime</code>	Created	<input checked="" type="checkbox"/>	Date the waiver was created.
<code>minorAgeLimitsInYears</code>	<code>int32</code>	Minor Age Limits in Years	<input checked="" type="checkbox"/>	What age a customer must be to be considered an adult when signing the waiver.
<code>name</code>	<code>string</code>	Name	<input checked="" type="checkbox"/>	Name of the waiver.
<code>requiresSignature</code>	<code>boolean</code>	Requires Signature	<input checked="" type="checkbox"/>	Whether or not the waiver requires a signature.
<code>terms</code>	<code>string</code>	Terms	<input checked="" type="checkbox"/>	Terms of the waiver.
<code>validityInDays</code>	<code>int32</code>	Validity in Days	<input checked="" type="checkbox"/>	How long the waiver is valid for once signed.
<code>waiverId</code>	<code>string</code>		<input checked="" type="checkbox"/>	Unique identifier of the waiver the customers are signing.

### 3.1.40 Webhooks

Catalog: `Roller`

Schema: `Data`

This is a read-only table. The Roller API may not support changing the data or the Invariantive SQL driver for Roller does not cover it. In the latter case, please use the table `NativePlatformScalarRequests` to upload data to the Roller API.

Select Roller API URL: `webhooks`

Insert Roller API URL: `webhooks`

Update Roller API URL: `webhooks`

Delete Roller API URL: `webhooks`

Field Selection Method: `NotRequired`

## Table Columns

The columns of the Table `webhooks` are shown below. Each column has an SQL data type.

Name	Data Type	Label	Required	Documentation
<code>createdDate</code>	<code>datetime</code>	Created	<input type="checkbox"/>	Date the staff record was created.
<code>webhookId</code>	<code>int32</code>		<input checked="" type="checkbox"/>	Webhook ID.

## 4 Schema: Native

### 4.1 Tables

#### 4.1.1 NATIVEPLATFORMSCALARREQUESTS: Roller Native Platform Scalar Requests

Direct access to native API.

Catalog: Roller

Schema: Native

Alias: npt

Label: Native Platform Scalar Requests

Documentation:

The NativePlatformScalarRequests table provides direct access to the native API protocol over an established connection to the Roller API server. It will contain a new row for every row inserted with a native API request in PAYLOAD\_TEXT with the results of unaltered forwarding of the payload to the Roller API server.

Retrieve: true

Insert: true

Update: false

Delete: false

### View Columns

The columns of the View NATIVEPLATFORMSCALARREQUESTS are shown below. Each column has an SQL data type. A new non-null value must be provided for every required column at all times during insert.

Name	Data Type	Label	Required	Documentation
BLOB_PREFERRED	boolean	BLOB Preferred	<input checked="" type="checkbox"/>	Indicator whether a BLOB result is preferred over text.
BOL_RESPONSE_CACHE_MAX_AGE_SEC	int32	Response Cache Maximum Age (sec)	<input type="checkbox"/>	Maximum age in seconds of Bridge Online response cache entries to be used.
CONTENT_TYPE	string(240)	Content Type	<input type="checkbox"/>	
DATE_ENDED	datetime	End Date	<input checked="" type="checkbox"/>	
DATE_STARTED	datetime	Start Date	<input checked="" type="checkbox"/>	
DRY_RUN	boolean	Run without Actions	<input checked="" type="checkbox"/>	
DURATION_MS	int32	Duration (ms)	<input checked="" type="checkbox"/>	
ERROR_MESSAGE_CODE	string(30)	Error Message Code	<input type="checkbox"/>	
ERROR_MESSAGE_TEXT	string(32000)	Error Message Text	<input type="checkbox"/>	
FAIL_ON_ERROR	boolean	Fail on Error	<input checked="" type="checkbox"/>	Whether to raise an exception when processing the native request triggered an error from the provider.
HTTP_DISK_CACHE_MAX_AGE_SEC	int32	HTTP Disk Cache Maximum Age (sec)	<input type="checkbox"/>	Maximum age in seconds of HTTP disk cache entries to be used.

Name	Data Type	Label	Required	Documentation
HTTP_DISK_CACHE_SAVE	boolean	Save HTTP Disk Cache	<input type="checkbox"/>	Whether results can be stored in HTTP disk cache.
HTTP_DISK_CACHE_USE	boolean	Use HTTP Disk Cache	<input type="checkbox"/>	Whether results can be fetched from HTTP disk cache.
HTTP_MEMORY_CACHE_MAX_AGE_SEC	int32	HTTP Memory Cache Maximum Age (sec)	<input type="checkbox"/>	Maximum age in seconds of HTTP memory cache entries to be used.
HTTP_MEMORY_CACHE_SAVE	boolean	Save HTTP Memory Cache	<input type="checkbox"/>	Whether results can be stored in HTTP memory cache.
HTTP_MEMORY_CACHE_USE	boolean	Use HTTP Memory Cache	<input type="checkbox"/>	Whether results can be fetched from HTTP memory cache.
HTTP_METHOD	string(30)	HTTP Method	<input type="checkbox"/>	
HTTP_STATUS_CODE	int16	HTTP Status Code	<input type="checkbox"/>	
ORIG_SYSTEM_GROUP	string(4000)	Original System Group	<input type="checkbox"/>	
ORIG_SYSTEM_REFERENCE	string(4000)	Original System Reference	<input type="checkbox"/>	
PAYLOAD_TEXT	string	Payload	<input type="checkbox"/>	
RESULT_BLOB	byte[]	Result BLOB	<input type="checkbox"/>	
RESULT_DATE_TIME_UTC	datetime		<input type="checkbox"/>	
RESULT_NUMBER	decimal		<input type="checkbox"/>	
RESULT_TEXT	string	Result Text	<input type="checkbox"/>	
SUCCESSFUL	boolean	Successful	<input checked="" type="checkbox"/>	
TIMEOUT_SEC	int32	Timeout (sec)	<input type="checkbox"/>	Timeout in seconds.
TRANSACTION_ID	int32	Transaction ID	<input checked="" type="checkbox"/>	Incrementing ID of the transaction.
URL	string(4000)	URL	<input type="checkbox"/>	

# Index

## - A -

Accept Marketing 28, 30  
 acceptMarketing 28, 30  
 Account Receivable 53  
 accountsReceivable 53  
 actualCashBalance 61  
 add-odata-mandatory-filters 2  
 addressCity 28  
 addressCountry 28  
 addressPostcode 28  
 addressState 28  
 addressStreet 28  
 addressSuburb 28  
 Allow Multiple Bookings 38  
 allowMultipleBookings 38  
 amount 42  
 amountOff 35  
 analysis-enforce-row-uniqueness 2  
 api-access-token 2  
 api-client-id 2  
 api-client-secret 2  
 api-pre-expiry-refresh-sec 2  
 api-redirect-url 2  
 api-refresh-token 2  
 api-scope 2  
 api-token-url 2  
 api-url 2  
 Attendance Location by Date 15  
 AttendanceLocationsByDate 15  
 Attendances by Date 16  
 AttendancesByDate 16  
 authorizingStaffId 24

## - B -

Balance 36  
 BLOB Preferred 65  
 BLOB\_PREFERRED 65  
 BOL\_RESPONSE\_CACHE\_MAX\_AGE\_SEC 65  
 Booking by ID 17  
 Booking Date 19, 21, 22, 59  
 Booking Items 19  
 Booking Items by Date 22  
 Booking Items by ID 21  
 Booking Payment ID 24, 53

Booking Payments by Date 24  
 Booking Reference 15, 16, 17, 21, 22, 24, 26, 27, 36, 38, 40, 41, 53  
 Booking Signed Waivers by Date 27  
 booking\_bookingReference 19  
 booking\_comments 21  
 booking\_companyId 21  
 booking\_createdDate 19, 21  
 booking\_customerId 19, 21  
 booking\_discount 21  
 booking\_externalId 21  
 booking\_fees 21  
 booking\_name 19, 21  
 booking\_remainder 21  
 booking\_source 21  
 booking\_status 19, 21  
 booking\_total 19, 21  
 booking\_uniqueId 19, 21  
 BookingById 17  
 bookingCreatedByStaffId 22  
 bookingCreatedDate 22  
 bookingCustomerId 15, 16, 22  
 bookingDate 19, 21, 22, 59  
 bookingDateRestrictionDateRangeDays 35  
 bookingDateRestrictionDateRangeEndDate 35  
 bookingDateRestrictionDteRangeStartDate 35  
 bookingDateRestrictionFromNumber 35  
 bookingDateRestrictionFromType 35  
 bookingEndDate 21  
 bookingItemId 21  
 bookingItemPartId 15, 16  
 BookingItems 19  
 BookingItemsByBookingId 21  
 BookingItemsByDate 22  
 bookingLocation 22  
 bookingModifiedDate 22  
 bookingPaymentId 24, 53  
 BookingPaymentsByDate 24  
 bookingReference 15, 16, 17, 21, 22, 24, 26, 27, 36, 38, 40, 41, 53  
 bookingReferences 38, 40, 41, 53  
 bookingRuleNumberOfUses 35  
 bookingRuleType 35  
 Bookings 26  
 BookingSignedWaiversByDate 27  
 bookingStatus 22  
 bulk-delete-page-size-rows 2  
 bulk-insert-page-size-bytes 2  
 bulk-insert-page-size-rows 2

**- C -**

capacityRemaining	44, 47, 49
cardRefunds	61
cardTakings	61
cashAdded	61
cashFloat	61
cashRefunds	61
cashRemoved	61
cashTakings	61
cashVariance	61
category_glCode	52
category_name	52
Check-in	15, 16
checkInDateTime	15, 16
checkTakings	61
Code Generation Mode	35
codeGenerationMode	35
codeOfConduct	62
comment	38
comments	17
companyId	17
Contact Number	30, 55, 57
contactNumber	30, 55, 57
Content Type	65
CONTENT_TYPE	65
cost	46, 50
Country	30, 62
countryCode	62
createdDate	17, 22, 24, 26, 27, 30, 31, 36, 38, 55, 57, 59, 63, 64
creditCardFeePercent	62
creditCardLast4Digits	24
creditValue	38
currency	62
Customer by ID	28
Customer ID	15, 16, 17, 22, 26, 30, 53, 55, 59
CustomerById	28
customerId	15, 16, 17, 26, 28, 30, 53, 55, 59
Customers by Date	30
CustomersByDate	30
customTicketId	41, 59
dateOfBirth	28, 30, 55
Deferred Revenue	53
deferredRevenue	53
deferredRevenueGiftCards	53
deferredRevenueOther	53
deviceId	24, 31, 61
deviceName	31
Devices	31
deviceStatus	31
deviceType	31
Discount	17, 53
Discount Amount	35
Discount Code	32, 53
Discount Codes	32
Discount ID	35, 58
Discount Products	33
discount_amountOff	32, 33
discount_bookingDateRestrictionDateRangeDays	32, 33
discount_bookingDateRestrictionDateRangeEndDate	32, 33
discount_bookingDateRestrictionDteRangeStartDate	32, 33
discount_bookingDateRestrictionFromNumber	32, 33
discount_bookingDateRestrictionFromType	32, 33
discount_bookingRuleNumberOfUses	32, 33
discount_bookingRuleType	32, 33
discount_codeGenerationMode	32, 33
discount_discountId	32, 33
discount_endDate	32, 33
discount_isSingleUseCode	32, 33
discount_maxApplicableAmount	32, 33
discount_name	32, 33
discount_percentOff	32, 33
discount_startDate	32, 33
discount_usageLimitNumberOfUses	32, 33
discount_usageLimitType	32, 33
discountAmount	22
discountCode	32, 53
DiscountCodes	32
discountId	35, 58
DiscountProducts	33
Discounts	35
download-error-400-bad-request-max-tries	2
download-error-400-bad-request-sleep-initial-ms	2
download-error-400-bad-request-sleep-max-ms	2
download-error-400-bad-request-sleep-multiplicator	2
download-error-408-request-timeout-max-tries	2
download-error-408-request-timeout-sleep-initial-ms	2

**- D -**

Database Driver	1
Date	19, 26, 38, 40, 41, 42, 44, 45, 46, 47, 49
Date of Birth	28, 30, 55
DATE_ENDED	65
DATE_STARTED	65

download-error-408-request-timeout-sleep-max-ms	2	download-error-argument-exception-sleep-initial-ms	2
download-error-408-request-timeout-sleep-multiplicat	2	download-error-argument-exception-sleep-max-ms	2
download-error-422-bad-request-max-tries	2	download-error-argument-exception-sleep-multiplicator	2
download-error-422-bad-request-sleep-initial-ms	2	download-error-internet-down-max-tries	2
download-error-422-bad-request-sleep-max-ms	2	download-error-internet-down-sleep-initial-ms	2
download-error-422-bad-request-sleep-multiplicator	2	download-error-internet-down-sleep-max-ms	2
download-error-429-too-many-requests-max-tries	2	download-error-internet-down-sleep-multiplicator	2
download-error-429-too-many-requests-sleep-initial-ms	2	download-error-io-exception-max-tries	2
download-error-429-too-many-requests-sleep-max-ms	2	download-error-io-exception-sleep-initial-ms	2
download-error-429-too-many-requests-sleep-multiplicat	2	download-error-io-exception-sleep-max-ms	2
download-error-502-server-unavailable-max-tries	2	download-error-io-exception-sleep-multiplicator	2
download-error-502-server-unavailable-sleep-initial-ms	2	download-error-json-exception-max-tries	2
download-error-502-server-unavailable-sleep-max-ms	2	download-error-json-exception-sleep-initial-ms	2
download-error-502-server-unavailable-sleep-multiplicat	2	download-error-json-exception-sleep-max-ms	2
download-error-503-server-unavailable-max-tries	2	download-error-json-exception-sleep-multiplicator	2
download-error-503-server-unavailable-sleep-initial-ms	2	download-error-other-exception-max-tries	2
download-error-503-server-unavailable-sleep-max-ms	2	download-error-other-exception-sleep-initial-ms	2
download-error-503-server-unavailable-sleep-multiplicat	2	download-error-other-exception-sleep-max-ms	2
download-error-504-gateway-timeout-max-tries	2	download-error-other-exception-sleep-multiplicator	2
download-error-504-gateway-timeout-sleep-initial-ms	2	download-error-socket-exception-max-tries	2
download-error-504-gateway-timeout-sleep-max-ms	2	download-error-socket-exception-sleep-initial-ms	2
download-error-504-gateway-timeout-sleep-multiplicat	2	download-error-socket-exception-sleep-max-ms	2
download-error-590-network-connect-timeout-max-tries	2	download-error-socket-exception-sleep-multiplicator	2
download-error-590-network-connect-timeout-sleep-initi	2	download-error-web-exception-max-tries	2
download-error-590-network-connect-timeout-sleep-max	2	download-error-web-exception-sleep-initial-ms	2
download-error-590-network-connect-timeout-sleep-mul	2	download-error-web-exception-sleep-max-ms	2
download-error-599-network-connect-timeout-max-tries	2	download-error-web-exception-sleep-multiplicator	2
download-error-599-network-connect-timeout-sleep-initi	2	download-error-web-not-implemented-max-tries	2
download-error-599-network-connect-timeout-sleep-max	2	download-error-web-not-implemented-sleep-initial-ms	2
download-error-599-network-connect-timeout-sleep-mul	2	download-error-web-not-implemented-sleep-max-ms	2
download-error-argument-exception-max-tries	2	download-error-web-not-implemented-sleep-multiplicat	2
		download-error-web-timeout-max-tries	2
		download-error-web-timeout-sleep-initial-ms	2
		download-error-web-timeout-sleep-max-ms	2
		download-error-web-timeout-sleep-multiplicator	2
		download-error-web-unauthorized-max-tries	2
		download-error-web-unauthorized-sleep-initial-ms	2
		download-error-web-unauthorized-sleep-max-ms	2
		download-error-web-unauthorized-sleep-multiplicator	2
		DRY_RUN	65
		Duration (ms)	65
		DURATION_MS	65

**- E -**

Email 28, 30, 55, 57  
 Employee ID 15, 16  
 employeeld 15, 16  
 End Date 35, 65  
 endDate 15, 16, 22, 24, 27, 30, 35, 36, 53, 55, 58, 59, 61  
 endDateDate 61  
 endTime 21, 49  
 Error Message Code 65  
 Error Message Text 65  
 ERROR\_MESSAGE\_CODE 65  
 ERROR\_MESSAGE\_TEXT 65  
 Event Date 41, 53  
 Event Type 53  
 eventDate 41, 53  
 eventType 53  
 expectedCashBalance 61  
 Expires 36, 55, 59  
 expiryDate 36, 55, 59  
 External Payment Reference 53  
 externalId 17  
 externalPaymentReference 53

**- F -**

Fail on Error 65  
 FAIL\_ON\_ERROR 65  
 Fee Revenue 53  
 feePerTransaction 62  
 feeRevenue 53  
 fees 17, 62  
 First Name 28, 30, 55, 57  
 firstName 28, 30, 55, 57  
 force-case-sensitive-identifiers 2  
 forced-casing-identifiers 2  
 Funds Received 53  
 fundsReceived 53

**- G -**

Gender 30  
 General Ledger Account Code 51, 52  
 Gift Card ID 36  
 Gift Card Number 36  
 Gift Cards by Date 36  
 giftCardId 36  
 giftCardNumber 36

GiftCardsByDate 36  
 giftCardTakings 61  
 glCode 51, 52  
 Group Size 22  
 groupSize 22

**- H -**

hqCode 50  
 hqProductId 50  
 HTTP Disk Cache Maximum Age (sec) 65  
 HTTP Memory Cache Maximum Age (sec) 65  
 HTTP Method 65  
 HTTP Status Code 65  
 HTTP\_DISK\_CACHE\_MAX\_AGE\_SEC 65  
 HTTP\_DISK\_CACHE\_SAVE 65  
 HTTP\_DISK\_CACHE\_USE 65  
 HTTP\_MEMORY\_CACHE\_MAX\_AGE\_SEC 65  
 HTTP\_MEMORY\_CACHE\_SAVE 65  
 HTTP\_MEMORY\_CACHE\_USE 65  
 HTTP\_METHOD 65  
 HTTP\_STATUS\_CODE 65  
 http-disk-cache-compression-level 2  
 http-disk-cache-directory 2  
 http-disk-cache-ignore-write-errors 2  
 http-disk-cache-max-age-sec 2  
 http-get-timeout-max-ms 2  
 http-get-timeout-ms 2  
 http-memory-cache-compression-level 2  
 http-memory-cache-max-age-sec 2  
 http-post-timeout-max-ms 2  
 http-post-timeout-ms 2

**- I -**

ignore-http-400-errors 2  
 ignore-http-401-errors 2  
 ignore-http-402-errors 2  
 ignore-http-403-errors 2  
 ignore-http-404-errors 2  
 ignore-http-422-errors 2  
 ignore-http-429-errors 2  
 ignore-http-500-errors 2  
 ignore-http-502-errors 2  
 ignore-http-503-errors 2  
 imageUrl 45, 46  
 Initial Value 36  
 initialValue 36  
 invalid-json-on-get-max-tries 2  
 invalid-json-on-get-sleep-initial-ms 2

- invalid-json-on-get-sleep-max-ms 2
- invalid-json-on-get-sleep-multiplicator 2
- invalid-json-on-post-max-tries 2
- invalid-json-on-post-sleep-initial-ms 2
- invalid-json-on-post-sleep-max-ms 2
- invalid-json-on-post-sleep-multiplicator 2
- invantive-sql-compress-sparse-arrays 2
- invantive-sql-correct-invalid-date 2
- invantive-sql-forward-filters-to-data-containers 2
- invantive-sql-share-byte-arrays 2
- invantive-sql-share-strings 2
- invantive-sql-shuffle-fetch-results-data-containers 2
- invantive-use-cache 2
- Is for Minor 55
- Is Single Use Code 35
- Is Valid 55
- isForMinor 55
- isSingleUseCode 35
- isValid 55
- itgen\_result\_date\_time 65
- itgen\_result\_number 65
- itgen\_rlr\_actualcashbalance 61
- itgen\_rlr\_addresscity 28
- itgen\_rlr\_addresscountry 28
- itgen\_rlr\_addresspostcode 28
- itgen\_rlr\_addressstate 28
- itgen\_rlr\_addressstreet 28
- itgen\_rlr\_addresssuburb 28
- itgen\_rlr\_amount 42
- itgen\_rlr\_authorizingstaffid 24
- itgen\_rlr\_booking\_bookingreference 19
- itgen\_rlr\_booking\_comments 21
- itgen\_rlr\_booking\_companyid 21
- itgen\_rlr\_booking\_createddate 19, 21
- itgen\_rlr\_booking\_customerid 19, 21
- itgen\_rlr\_booking\_discount 21
- itgen\_rlr\_booking\_externalid 21
- itgen\_rlr\_booking\_fees 21
- itgen\_rlr\_booking\_name 19, 21
- itgen\_rlr\_booking\_remainder 21
- itgen\_rlr\_booking\_source 21
- itgen\_rlr\_booking\_status 19, 21
- itgen\_rlr\_booking\_total 19, 21
- itgen\_rlr\_booking\_uniqueid 19, 21
- itgen\_rlr\_bookingcreatedbystaffid 22
- itgen\_rlr\_bookingcreateddate 22
- itgen\_rlr\_bookingdaterestrictiondaterangedays 35
- itgen\_rlr\_bookingdaterestrictiondaterangeenddate 35
- itgen\_rlr\_bookingdaterestrictiondterangestartdate 35
- itgen\_rlr\_bookingdaterestrictionfromnumber 35
- itgen\_rlr\_bookingdaterestrictionfromtype 35
- itgen\_rlr\_bookingenddate 21
- itgen\_rlr\_bookingitemid 21
- itgen\_rlr\_bookingitempartid 15, 16
- itgen\_rlr\_bookingrulenumeroofuses 35
- itgen\_rlr\_bookingruletype 35
- itgen\_rlr\_capacityremaining 44, 47, 49
- itgen\_rlr\_cardrefunds 61
- itgen\_rlr\_cardtakings 61
- itgen\_rlr\_cashadded 61
- itgen\_rlr\_cashfloat 61
- itgen\_rlr\_cashrefunds 61
- itgen\_rlr\_cashremoved 61
- itgen\_rlr\_cashtakings 61
- itgen\_rlr\_cashvariance 61
- itgen\_rlr\_category\_glcode 52
- itgen\_rlr\_category\_name 52
- itgen\_rlr\_checktakings 61
- itgen\_rlr\_codeofconduct 62
- itgen\_rlr\_comment 38
- itgen\_rlr\_comments 17
- itgen\_rlr\_companyid 17
- itgen\_rlr\_cost 46, 50
- itgen\_rlr\_countrycode 62
- itgen\_rlr\_creditcardfeepercent 62
- itgen\_rlr\_creditcardlast4digits 24
- itgen\_rlr\_creditvalue 38
- itgen\_rlr\_currency 62
- itgen\_rlr\_customticketid 41, 59
- itgen\_rlr\_date 42, 44, 49
- itgen\_rlr\_deferredrevenuegiftcards 53
- itgen\_rlr\_deferredrevenueother 53
- itgen\_rlr\_description 45, 46
- itgen\_rlr\_deviceid 24, 31, 61
- itgen\_rlr\_devicename 31
- itgen\_rlr\_devicestatus 31
- itgen\_rlr\_devicetype 31
- itgen\_rlr\_discount\_amountoff 32, 33
- itgen\_rlr\_discount\_bookingdaterestrictiondaterangedays 32, 33
- itgen\_rlr\_discount\_bookingdaterestrictiondaterangeenddate 32, 33
- itgen\_rlr\_discount\_bookingdaterestrictiondterangestartdate 32, 33
- itgen\_rlr\_discount\_bookingdaterestrictionfromnumber 32, 33
- itgen\_rlr\_discount\_bookingdaterestrictionfromtype 32, 33
- itgen\_rlr\_discount\_bookingrulenumeroofuses 32, 33
- itgen\_rlr\_discount\_bookingruletype 32, 33
- itgen\_rlr\_discount\_codegenerationmode 32, 33
- itgen\_rlr\_discount\_discountid 32, 33



log-native-calls-to-disk-on-success 2  
log-native-calls-to-trace 2

## - M -

maxApplicableAmount 35  
Maximum Applicable Amount 35  
maximum-length-identifiers 2  
max-odata-filters 2  
max-url-length-accepted 2  
max-url-length-desired 2  
Membership Redemptions 40  
Membership Statuses 38, 41  
MembershipCreditsByDate 38  
MembershipRedemptionsByDate 40  
MembershipStatusesByDate 41  
metadata-cache-max-age-sec 2  
Minor Age Limits in Years 63  
minorAgeLimitsInYears 63  
modifiedDate 27, 30, 36, 38, 55  
modifierGroupId 42  
modifierGroupName 42  
modifierId 42  
modifierName 42  
Modifiers 42  
modifierType 42

## - N -

Name 17, 26, 35, 38, 45, 46, 49, 50, 51, 52, 59, 62, 63  
Native Platform Scalar Requests 65  
NATIVEPLATFORMSCALARREQUESTS 65  
Net Revenue 53  
netRevenue 53  
nextStatus 41  
npt 65  
Number of Recurring Payments 59  
numberOfRecurringPayments 59

## - O -

oauth-unauthorized-max-tries 2  
oauth-unauthorized-sleep-initial-ms 2  
oauth-unauthorized-sleep-max-ms 2  
oauth-unauthorized-sleep-multiplicator 2  
onlineSalesOpen 42, 44, 49  
ORIG\_SYSTEM\_GROUP 65  
ORIG\_SYSTEM\_REFERENCE 65  
Original System Group 65

Original System Reference 65

## - P -

Parent Location ID 38  
Parent Product ID 15, 16, 50  
Parent Signed Waiver ID 55  
parent\_glCode 52  
parent\_name 52  
parentLocationId 38  
parentProductId 15, 16, 50  
parentSignedWaiverId 55  
partition-slot-based-rate-limit-length-ms 2  
partition-slot-based-rate-limit-slots 2  
Payload 65  
PAYLOAD\_TEXT 65  
Payment Type 53  
paymentSettingsApiUrl 62  
paymentSettingsConfigurationId 62  
paymentSettingsIntegrationId 62  
paymentType 53  
Percent Discount 35  
percentOff 35  
phone 28  
postcode 30  
pre-request-delay-ms 2  
previousStatus 41  
Product Availabilities by Date 42  
Product Availability by Date 45  
Product Availability Product Session Allocations by Date 44, 47  
Product Availability Product Sessions by Date 49  
Product ID 15, 16, 19, 21, 22, 33, 44, 47, 50, 52, 53, 59  
Product Name 15, 16  
Product Sub-type 50, 59  
Product Type 50, 53, 59  
ProductAvailabilitiesByDate 42  
productavailability\_description 42, 44, 46, 47, 49  
productavailability\_id 42, 44, 46, 47, 49  
productavailability\_imageUrl 42, 44, 46, 47, 49  
productavailability\_name 42, 44, 46, 47, 49  
productavailability\_type 42, 44, 46, 47, 49  
ProductAvailabilityAllocationsByDate 44  
ProductAvailabilityByDate 45  
ProductAvailabilityProductsByDate 46  
ProductAvailabilitySessionAllocationsByDate 47  
ProductAvailabilitySessionsByDate 49  
ProductCategory 42, 44, 45, 46, 47, 49  
productId 15, 16, 19, 21, 22, 33, 44, 47, 50, 52, 53, 59

ProductIds 19, 26, 42, 44, 45, 46, 47, 49  
 productName 15, 16  
 productPricesIncludeTax 62  
 Products 50  
 productStatus 50  
 productSubType 50, 59  
 productType 50, 53, 59

## - Q -

Quantity 19, 21, 22

## - R -

Receipt Number 15, 16, 24, 27, 53  
 receiptNumber 15, 16, 24, 27, 53  
 recognisedDiscount 53  
 Recognized Discount 53  
 Recurring Payment Frequency 59  
 recurringPaymentFrequency 59  
 redemptionDate 40  
 remainder 17  
 Reporting Categories 51, 52  
 Reporting Category Name 50  
 Reporting Products 52  
 ReportingCategories 51  
 ReportingCategoryCategories 52  
 reportingCategoryName 50  
 ReportingCategoryProducts 52  
 requested-page-size 2  
 requests-parallel-max 2  
 Requires Signature 63  
 requiresSignature 63  
 Response Cache Maximum Age (sec) 65  
 Result BLOB 65  
 Result Text 65  
 RESULT\_BLOB 65  
 RESULT\_DATE\_TIME\_UTC 65  
 RESULT\_NUMBER 65  
 RESULT\_TEXT 65  
 Revenues 53  
 role 57  
 Roller 1, 15, 16, 17, 19, 21, 22, 24, 26, 27, 28, 30, 31, 32, 33, 35, 36, 38, 40, 41, 42, 44, 45, 46, 47, 49, 50, 51, 52, 53, 55, 57, 58, 59, 61, 62, 63, 64, 65  
 Run without Actions 65

## - S -

Save HTTP Disk Cache 65

Save HTTP Memory Cache 65  
 Session End 15, 16, 22  
 Session Start 15, 16, 22  
 session\_capacityRemaining 47  
 session\_date 47  
 session\_endTime 47  
 session\_name 47  
 session\_onlineSalesOpen 47  
 session\_startTime 47  
 sessionEnd 15, 16, 22  
 sessionStart 15, 16, 22  
 Signed Waiver ID 27, 55  
 Signed Waivers by Date 55  
 signedWaiverId 27, 55  
 SignedWaiversByDate 55  
 simulate-http-400-errors 2  
 simulate-http-400-errors-percentage 2  
 simulate-http-401-errors 2  
 simulate-http-401-errors-percentage 2  
 simulate-http-403-errors 2  
 simulate-http-403-errors-percentage 2  
 simulate-http-408-errors 2  
 simulate-http-408-errors-percentage 2  
 simulate-http-429-errors 2  
 simulate-http-429-errors-percentage 2  
 simulate-http-500-errors 2  
 simulate-http-500-errors-percentage 2  
 simulate-http-502-errors 2  
 simulate-http-502-errors-percentage 2  
 simulate-http-503-errors 2  
 simulate-http-503-errors-percentage 2  
 simulate-http-protocol-errors 2  
 simulate-http-protocol-errors-percentage 2  
 simulate-http-timeout-errors 2  
 simulate-http-timeout-errors-percentage 2  
 slot-based-rate-limit-length-ms 2  
 slot-based-rate-limit-slots 2  
 source 17  
 Staff ID 24, 57  
 Staff Members 57  
 staffId 24, 57  
 StaffMembers 57  
 standardize-identifiers 2  
 standardize-identifiers-casing 2  
 Start Date 35, 61, 65  
 startDate 15, 16, 22, 24, 27, 30, 35, 36, 53, 55, 58, 59, 61  
 startTime 19, 21, 26, 49  
 status 17, 22, 26, 38  
 Successful 65  
 SUCCESSFUL 65

**- T -**

Tax on Funds Received 53  
Tax Payable 53  
Tax Percentage 53  
taxExemptId 22  
taxOnFundsReceived 53  
taxPayable 53  
taxPercent 53  
Terms 63  
Ticket Discounts by Date 58  
Ticket ID 27, 36, 38, 40, 41, 53, 59  
Ticket Transaction Value 53  
Ticket Unit Cost 53  
ticket\_bookingDate 58  
ticket\_createdDate 58  
ticket\_customerId 58  
ticket\_customTicketId 58  
ticket\_expiryDate 58  
ticket\_name 58  
ticket\_numberOfRecurringPayments 58  
ticket\_productId 58  
ticket\_productSubType 58  
ticket\_productType 58  
ticket\_recurringPaymentFrequency 58  
ticket\_ticketId 58  
TicketDiscountsByDate 58  
ticketId 27, 36, 38, 40, 41, 53, 59  
Tickets by Date 59  
TicketsByDate 59  
ticketTransactionValue 53  
ticketUnitCost 53  
TillReconciliations 61  
Timeout (sec) 65  
TIMEOUT\_SEC 65  
timeZone 62  
Tip 24  
Total 17, 24, 26  
Transaction Date 53  
Transaction Fee Amount 24, 53  
Transaction ID 24, 65  
Transaction Location 53  
TRANSACTION\_ID 65  
transactionDate 53  
transactionFeeAmount 24, 53  
transactionFeePercent 62  
transactionId 24  
transactionLocation 53  
type 45, 46

**- U -**

uniqueId 17, 26  
uniqueIdOrBookingId 17, 21  
URL 65  
usageLimitNumberOfUses 35  
usageLimitType 35  
Use HTTP Disk Cache 65  
Use HTTP Memory Cache 65  
use-batch-insert 2  
use-http-disk-cache-read 2  
use-http-disk-cache-write 2  
use-http-memory-cache-read 2  
use-http-memory-cache-write 2  
use-test-environment 2

**- V -**

Validity in Days 63  
validityInDays 63  
Venues 62

**- W -**

waiverId 55, 63  
Waivers 63  
webhookId 64  
Webhooks 64

**- Z -**

ZIP Code 30



# *invantive* the **SQL** company

Invantive B.V.  
Biesteweg 11  
3849 RD Hierden  
the Netherlands

Tel: +31 88 00 26 500  
Fax: +31 84 22 58 178  
info@invantive.com  
invantive.com

IBAN NL25 BUNQ 2098 2586 07  
Chamber of Industry and Commerce  
13031406  
VAT NL812602377B01  
RSIN 8122602377  
Managing Director: Guido Leenders  
Registered office: Roermond