Roller API Data Model

for use with Invantive SQL

24.0

invantive
the SQL company

Copyright

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 an 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

1 SQL Driver for Roller API

Invantive UniversalSQL 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. Invantive Support or other users will try to help you.

Online software for attractions, entertainment and leisure venues.

The Roller driver covers 41 tables and 452 columns.

Roller API Clients

Invantive UniversalSQL is available on many user interfaces ("clients" in traditional server-client paradigma). All Invantive UniversalSQL 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 UniversalSQL 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-premises 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 75 other (cloud) platforms. Invantive 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 UniversalSQL 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 UniversalSQL will create log files per Roller 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 UniversalSQL such as databases, data containers and partitioning can be found in the Invantive UniversalSQL grammar.

The configuration can be changed using various attributes from the database definition, on log on and during 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 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 Roller on the native connection 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.

2 {res:itgen_doc_sql_attr_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.
- 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. The reference manuals contain instructions how to relocate the settings*.xml files. Settings*.xml files are typically located in the `%USERPROFILE%\invantive` folder in most deployment scenarios. 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 Drivers 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	Enforce rows to be unique for software analysis. A fingerprint is calculated from the whole row of data when the primary key column is unknown.	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 authorization scope(s) to request an OAuth token for.	OData		✓		✓	
api-token-url	The token URI is the OAuth2 endpoint to exchange tokens with.	OData		✓		✓	
api-url	URL of web service.	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 HTTP server reports bad format during retrieval of data.		3	✓	✓	✓	
download-error-400-bad-request-sleep-initial-ms	Initial sleep in milliseconds between retries when HTTP 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 HTTP 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 HTTP server reports that the API server is unavailable during retrieval of data.		2	✓	✓	✓	
download-error-403-forbidden-max-tries	Maximum number of tries when the website reports a HTTP status 403 (Forbidden).		2	✓	✓	✓	
download-error-403-forbidden-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 403 (Forbidden).		3000	✓	✓	✓	
download-error-403-forbidden-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 403 (Forbidden).		10000	✓	✓	✓	
download-error-403-forbidden-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 403 (Forbidden).		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	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
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 HTTP server reports unprocessable entity during retrieval of data.		30	✓	✓	✓	
download-error-422-bad-request-sleep-initial-ms	Initial sleep in milliseconds between retries when HTTP server reports unprocessable entity during retrieval of data.		10000	✓	✓	✓	
download-error-422-bad-request-sleep-max-ms	Maximum sleep in milliseconds between retries when HTTP server reports unprocessable entity during retrieval of data.		300000	✓	✓	✓	
download-error-422-bad-request-sleep-multiplicator	Multiplication factor for sleep between retries HTTP server reports unprocessable entity during retrieval of data.		2	✓	✓	✓	
download-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	✓	✓	✓	
download-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	✓	✓	✓	
download-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	✓	✓	✓	
download-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	✓	✓	✓	
download-error-500-internal-server-error-max-tries	{res:itgen_pae_download_error_500_internal_server_error_max_tries}		10	✓	✓	✓	
download-error-500-internal-server-error-sleep-initial-ms	{res:itgen_pae_download_error_500_internal_server_error_sleep_initial_ms}		10000	✓	✓	✓	
download-error-500-internal-server-error-sleep-max-ms	{res:itgen_pae_download_error_500_internal_server_error_sleep_max_ms}		300000	✓	✓	✓	
download-error-500-internal-server-error-sleep-multiplicator	{res:itgen_pae_download_error_500_internal_server_error_sleep_multiplicator}		2	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
download-error-502-server-unavailable-max-tries	Maximum number of tries when HTTP server reports a bad gateway during retrieval of data.		30	✓	✓	✓	
download-error-502-server-unavailable-sleep-initial-ms	Initial sleep in milliseconds between retries when HTTP server reports a bad gateway during retrieval of data.		10000	✓	✓	✓	
download-error-502-server-unavailable-sleep-max-ms	Maximum sleep in milliseconds between retries when HTTP server reports that a bad gateway during retrieval of data.		300000	✓	✓	✓	
download-error-502-server-unavailable-sleep-multiplicator	Multiplication factor for sleep between retries HTTP server reports a bad gateway during retrieval of data.		2	✓	✓	✓	
download-error-503-server-unavailable-max-tries	Maximum number of tries when HTTP server reports that the API server is unavailable during retrieval of data.		30	✓	✓	✓	
download-error-503-server-unavailable-sleep-initial-ms	Initial sleep in milliseconds between retries when HTTP server reports that the API server is unavailable during retrieval of data.		10000	✓	✓	✓	
download-error-503-server-unavailable-sleep-max-ms	Maximum sleep in milliseconds between retries when HTTP server reports that the API server is unavailable during retrieval of data.		300000	✓	✓	✓	
download-error-503-server-unavailable-sleep-multiplicator	Multiplication factor for sleep between retries HTTP server reports that the API server is unavailable during retrieval of data.		2	✓	✓	✓	
download-error-504-gateway-timeout-max-tries	Maximum number of tries when the website reports a gateway timeout.		10	✓	✓	✓	
download-error-504-gateway-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a gateway timeout.		10000	✓	✓	✓	
download-error-504-gateway-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a gateway timeout.		300000	✓	✓	✓	
download-error-504-gateway-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a gateway timeout.		2	✓	✓	✓	
download-error-590-network-connect-timeout-max-tries	Maximum number of tries when the website reports a HTTP status 590.		10	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
download-error-590-network-connect-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 590.		10000	✓	✓	✓	
download-error-590-network-connect-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 590.		300000	✓	✓	✓	
download-error-590-network-connect-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 590.		2	✓	✓	✓	
download-error-599-network-connect-timeout-max-tries	Maximum number of tries when the website reports a HTTP status 599.		10	✓	✓	✓	
download-error-599-network-connect-timeout-sleep-initial-ms	Initial sleep in milliseconds between retries when the website reports a HTTP status 599.		10000	✓	✓	✓	
download-error-599-network-connect-timeout-sleep-max-ms	Maximum sleep in milliseconds between retries when the website reports a HTTP status 599.		300000	✓	✓	✓	
download-error-599-network-connect-timeout-sleep-multiplicator	Multiplication factor for sleep between retries when the website reports a HTTP status 599.		2	✓	✓	✓	
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-down-max-tries	Maximum number of tries when the Internet connection seems down during retrieval of data.		10	✓	✓	✓	
download-error-internet-down-sleep-initial-ms	Initial sleep in milliseconds between retries when the Internet connection seems down during retrieval of data.		10000	✓	✓	✓	
download-error-internet-down-	Maximum sleep in milliseconds between retries when the Internet		300000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
sleep-max-ms	connection seems down during retrieval of data.						
download-error-internet-down-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	✓	✓	✓	
download-error-json-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when an invalid JSON body is returned.		10000	✓	✓	✓	
download-error-json-exception-sleep-multiplicator	Multiplication factor for sleep between retries when an invalid JSON body is returned.		2	✓	✓	✓	
download-error-name-resolution-failure-max-tries	Maximum number of tries when the host name could not be resolved during retrieval of data.		5	✓	✓	✓	
download-error-name-resolution-failure-sleep-initial-ms	Initial sleep in milliseconds between retries when the host name could not be resolved during retrieval of data.		5000	✓	✓	✓	
download-error-name-resolution-failure-sleep-max-ms	Maximum sleep in milliseconds between retries when the host name could not be resolved during retrieval of data.		5000	✓	✓	✓	
download-error-name-resolution-failure-sleep-multiplicator	{res:itgen_pae_download_error_iname_resolution_failure_sleep_multiplicator}		1	✓	✓	✓	
download-error-other-exception-max-tries	Maximum number of tries when an unqualified error occurs during retrieval of data.		3	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
download-error-other-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when an unqualified error ocurrs during retrieval of data.		10000	✓	✓	✓	
download-error-other-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when an unqualified error ocurrs during retrieval of data.		300000	✓	✓	✓	
download-error-other-exception-sleep-multiplicator	Multiplication factor for sleep between retries when an unqualified error ocurrs during retrieval of data.		2	✓	✓	✓	
download-error-socket-exception-max-tries	Maximum number of tries when the network connection is forcible dropped during retrieval of data.		10	✓	✓	✓	
download-error-socket-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when the network connection is forcible dropped during retrieval of data.		10000	✓	✓	✓	
download-error-socket-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when the network connection is forcible dropped during retrieval of data.		300000	✓	✓	✓	
download-error-socket-exception-sleep-multiplicator	Multiplication factor for sleep between retries when the network connection is forcible dropped during retrieval of data.		2	✓	✓	✓	
download-error-web-exception-max-tries	Maximum number of tries when a web connection failure occurs during retrieval of data.		10	✓	✓	✓	
download-error-web-exception-sleep-initial-ms	Initial sleep in milliseconds between retries when a web connection failure occurs during retrieval of data.		10000	✓	✓	✓	
download-error-web-exception-sleep-max-ms	Maximum sleep in milliseconds between retries when a web connection failure occurs during retrieval of data.		300000	✓	✓	✓	
download-error-web-exception-sleep-multiplicator	Multiplication factor for sleep between retries when a web connection failure occurs during retrieval of data.		2	✓	✓	✓	
download-error-web-not-found-max-tries	{res:itgen_pae_download_error_web_not_found_max_tries}		1	✓	✓	✓	
download-error-web-not-found-sleep-initial-ms	{res:itgen_pae_download_error_web_not_found_sleep_initial_ms}		10000	✓	✓	✓	
download-error-web-not-found-sleep-max-ms	{res:itgen_pae_download_error_web_not_found_sleep_max_ms}		300000	✓	✓	✓	
download-error-web-not-found-sleep-multiplicator	{res:itgen_pae_download_error_web_not_found_sleep_multiplicator}		2	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
download-error-web-not-implemented-max-tries	Maximum number of tries when the connection reports not implemented.		1	✓	✓	✓	
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	✓	✓	✓	
download-error-web-unknown-max-tries	{res:itgen_pae_download_error_web_unknown_max_tries}		10	✓	✓	✓	
download-error-web-unknown-sleep-initial-ms	{res:itgen_pae_download_error_web_unknown_sleep_initial_ms}		5000	✓	✓	✓	
download-error-web-unknown-sleep-max-ms	{res:itgen_pae_download_error_web_unknown_sleep_max_ms}		30000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
download-error-web-unknown-sleep-multiplicator	{res:itgen_pae_download_error_web_unknown_sleep_multiplicator}		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.		5	✓	✓	✓	
http-disk-cache-directory	Directory where HTTP cache is stored.		C:\Users\guido\Invantive\Cache\http\guido\shared	✓	✓	✓	
http-disk-cache-ignore-write-errors	Whether to ignore write errors to disk cache.		False	✓	✓	✓	
http-disk-cache-max-age-sec	Maximum acceptable age in seconds for use of data in the HTTP disk cache.		2592000	✓	✓	✓	
http-get-timeout-max-ms	HTTP GET maximum timeout on retry (ms).		24000	✓	✓	✓	
http-get-timeout-ms	HTTP GET timeout (ms).		56000	✓	✓	✓	
http-memory-cache-compression-level	Compression level for the HTTP memory cache, ranging from 1 (little) to 9 (intense). Default is 5.		5	✓	✓	✓	
http-memory-cache-max-age-sec	Maximum acceptable age in seconds for use of data in the HTTP memory cache.		14400	✓	✓	✓	
http-post-timeout-max-ms	HTTP POST maximum timeout on retry (ms).		58000	✓	✓	✓	
http-post-timeout-ms	HTTP POST timeout (ms).		57000	✓	✓	✓	
http-public-disk-cache-directory	{res:itgen_pae_http_public_disk_cache_directory}		C:\Users\guido\Invantive\Cache\public	✓	✓	✓	
ignore-http-400-errors	Ignore HTTP 400 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-401-errors	Ignore HTTP 401 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-402-errors	Ignore HTTP 402 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-403-errors	Ignore HTTP 403 errors when exchanging results with the HTTP		False	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
	endpoint.						
ignore-http-404-errors	Ignore HTTP 404 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-422-errors	Ignore HTTP 422 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-429-errors	Ignore HTTP 429 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-500-errors	Ignore HTTP 500 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-502-errors	Ignore HTTP 502 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
ignore-http-503-errors	Ignore HTTP 503 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
invalid-json-on-get-max-tries	Maximum number of tries when the JSON received on GET is invalid.		1	✓	✓	✓	
invalid-json-on-get-sleep-initial-ms	Initial sleep in milliseconds between retries when the JSON received on GET is invalid.		1000	✓	✓	✓	
invalid-json-on-get-sleep-max-ms	Maximum sleep in milliseconds between retries when the JSON received on GET is invalid.		10000	✓	✓	✓	
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.		1000	✓	✓	✓	
invalid-json-on-post-sleep-max-ms	Maximum sleep in milliseconds between retries when the JSON received on POST is invalid.		10000	✓	✓	✓	
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	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
invantive-sql-execution-profile-disk-path	{res:itgen_pae_invantive_sql_execution_profile_disk_path}	SQL Engine V1		✓	✓	✓	
invantive-sql-execution-profile-to-disk	{res:itgen_pae_invantive_sql_execution_profile_to_disk}	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	✓	✓	✓	
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 call events to register from last activation.	Shared		✓	✓	✓	
log-native-calls-to-disk-max-seconds	Maximum number of seconds to register calls from last activation.	Shared		✓	✓	✓	
log-native-calls-to-disk-on-error	Registers native calls to data container backend as disk files when the call raised an error.	Shared	False	✓	✓	✓	
log-native-calls-to-disk-on-success	Registers native calls to data container backend as disk files when the call raised no error.	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	Maximum number of OData filter elements.	OData	100	✓	✓	✓	
max-odata-rewrite-in-count	{res:itgen_pae_max_odata_rewrite_in_count}	OData	500	✓	✓	✓	
max-url-length-accepted	The maximum accepted URL length before raising an error.		8000	✓	✓	✓	
max-url-length-desired	The maximum desired URL length.		8000	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
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.		2	✓	✓	✓	
oauth-unauthorized-sleep-initial-ms	Initial sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails.		10000	✓	✓	✓	
oauth-unauthorized-sleep-max-ms	Maximum sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails.		1000	✓	✓	✓	
oauth-unauthorized-sleep-multiplicator	Multiplication factor for sleep between OAuth reauthentication tries when the OAuth authentication fails.		2	✓	✓	✓	
partition-slot-based-rate-limit-length-ms	Total length in milliseconds 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-maximum-usable-partitions	Requested Maximum Usable Partitions	Shared		✓	✓	✓	
requested-page-size	Preferred number of rows to exchange per round trip; only effective on limited platforms such as AFAS Online.	Shared		✓	✓	✓	
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 HTTP endpoint.		False	✓	✓	✓	
simulate-http-400-errors-percentage	Percentage of simulated HTTP 400 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-401-errors	Simulate HTTP 401 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-401-errors-percentage	Percentage of simulated HTTP 401 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-403-errors	Simulate HTTP 403 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-403-errors-percentage	Percentage of simulated HTTP 403 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
simulate-http-408-errors	Simulate HTTP 408 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-408-errors-percentage	Percentage of simulated HTTP 408 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-429-errors	Simulate HTTP 429 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-429-errors-percentage	Percentage of simulated HTTP 429 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-500-errors	Simulate HTTP 500 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-500-errors-percentage	Percentage of simulated HTTP 500 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-502-errors	Simulate HTTP 502 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-502-errors-percentage	Percentage of simulated HTTP 502 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-503-errors	Simulate HTTP 503 errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-503-errors-percentage	Percentage of simulated HTTP 503 errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
simulate-http-504-errors	{res:itgen_pae_simulate_http_504_errors}		False	✓	✓	✓	
simulate-http-504-errors-percentage	{res:itgen_pae_simulate_http_504_errors_percentage}		0	✓	✓	✓	
simulate-http-522-errors	{res:itgen_pae_simulate_http_522_errors}		False	✓	✓	✓	
simulate-http-522-errors-percentage	{res:itgen_pae_simulate_http_522_errors_percentage}		0	✓	✓	✓	
simulate-http-524-errors	{res:itgen_pae_simulate_http_524_errors}		False	✓	✓	✓	
simulate-http-524-errors-percentage	{res:itgen_pae_simulate_http_524_errors_percentage}		0	✓	✓	✓	
simulate-http-protocol-errors	Simulate HTTP protocol errors when exchanging results with the HTTP endpoint.		False	✓	✓	✓	
simulate-http-protocol-errors-percentage	Percentage of simulated HTTP protocol errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	

Code	Description	Origin	Default Value	Set from Connection String	Set from Set SQL-Statement	Set from Drivers File	Set from Log On
simulate-http-timeout-errors	Simulate HTTP timeout errors when exchanging results with the HTTP endpoint..		False	✓	✓	✓	
simulate-http-timeout-errors-percentage	Percentage of simulated HTTP timeout errors when exchanging results with the HTTP endpoint.		0	✓	✓	✓	
slot-based-rate-limit-length-ms	Total length in milliseconds 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	✓	✓	✓	
totp-secret	Shared secret key to generate one-time password using TOTP RFC 6238. For improved security, manually enter the one-time password asked during login.	OData		✓		✓	✓
use-batch-insert	Whether to use batch insert.	OData	True	✓	✓	✓	
use-http-disk-cache	Combination of use-http-disk-cache-read and use-http-disk-cache-write.			✓	✓	✓	
use-http-disk-cache-read	Whether to use HTTP responses from previous queries stored on disk to answer the current query.		False	✓	✓	✓	
use-http-disk-cache-write	Whether to memorize HTTP responses on disk.		False	✓	✓	✓	
use-http-memory-cache	Combination of use-http-memory-cache-read and use-http-memory-cache-write.			✓	✓	✓	
use-http-memory-cache-read	Whether to use HTTP responses from previous queries stored in memory that can answer the current query.		True	✓	✓	✓	
use-http-memory-cache-write	Whether to memorize HTTP responses from previous queries for use by future queries.		True	✓	✓	✓	
use-test-environment	Use the test environment. If false or null, the production environment will be used.	Roller	False	✓		✓	✓

3 Catalog: Roller
3.1 Schemas
3.1.1 Schema: Data
3.1.1.1 Tables
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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
endDate		☑		Search end date, based on record modified date.
startDate		☑		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	varchar2	Customer ID	☐	Unique identifier of the customer linked to the booking.
bookingItemPartId	varchar2		☑	Unique identifier for a ticket.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
checkInDateTime	datetime	Check-in	☑	Date/time of the attendance.

Name	Data Type	Label	Required	Documentation
customerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the specific ticket.
employeeId	varchar2	Employee ID	☐	The identifier of the staff member w ho completed the check in.
locationId	varchar2		☑	Location the ticket operates in the venue.
parentProductId	varchar2	Parent Product ID	☑	Unique identifier of the parent product being checked in.
productId	varchar2	Product ID	☑	Unique identifier of the product being checked in.
productName	varchar2	Product Name	☑	
receiptNumber	varchar2	Receipt Number	☑	
sessionEnd	varchar2	Session End	☐	End time of the ticket if applicable.
sessionStart	varchar2	Session Start	☐	Start time of the ticket if applicable.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.

Name	Data Type	Required	Default Value	Documentation
startDate		☑		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	varchar2	Customer ID	☐	Unique identifier of the customer linked to the booking.
bookingItemPartId	varchar2		☑	Unique identifier for a ticket.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
checkInDateTime	datetime	Check-in	☑	Date/time of the attendance.
customerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the specific ticket.
employeeId	varchar2	Employee ID	☐	The identifier of the staff member w ho completed the check in.
parentProductId	varchar2	Parent Product ID	☑	Unique identifier of the parent product being checked in.
productId	varchar2	Product ID	☑	Unique identifier of the product being checked in.
productName	varchar2	Product Name	☑	
receiptNumber	varchar2	Receipt Number	☑	
sessionEnd	varchar2	Session End	☐	End time of the ticket if applicable.
sessionStart	varchar2	Session Start	☐	Start time of the ticket if applicable.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
uniqueIdOrBookingId		☐		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
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
comments	varchar2		☐	Additional information about the booking.
companyId	varchar2		☐	The Id of the company who made the booking.
createdDate	datetime	Created	☑	Date the booking item was created.
customerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the booking.
discount	decimal	Discount	☐	Total discount applied.
externalId	varchar2		☐	External booking ID.
fees	decimal		☐	Total fees applied.
name	varchar2	Name	☐	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
remainder	decimal		☐	Total remaining cost/outstanding balance.
source	varchar2		☐	The application that created the booking.
status	varchar2		☐	Booking status.
total	decimal	Total	☐	Total cost.
uniqueID	guid		☐	Unique booking ID.

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 UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐		Date of the bookings you are looking for.
keywords		☐		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.
locationIds		☐		Comma separated capacity location IDs. Date is required.
productIds		☐		Comma separated product IDs. Date is required.
startTime		☐		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	varchar2		☑	Unique identifier for a booking.
booking_createdDate	datetime		☑	Date the booking item w as created.
booking_customerId	varchar2		☐	Unique identifier of the customer linked to the booking.
booking_name	varchar2		☐	Name of the booking can be different to the booking ow ner (defaults to the booking ow ner's name).
booking_status	varchar2		☐	Booking status.
booking_total	decimal		☐	Total cost.
booking_uniqueID	guid		☐	Unique booking ID.
bookingDate	datetime	Booking Date	☐	The date this item w as purchased or the date this item is valid for.
productId	int32	Product ID	☐	Unique identifier for a product.
quantity	int32	Quantity	☐	Number of items purchased - same as number of tickets except w hen GroupSize is greater than one.
startTime	varchar2		☐	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐		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	varchar2		☐	Additional information about the booking.
booking_companyId	varchar2		☐	The Id of the company who made the booking.
booking_createdDate	datetime		☑	Date the booking item was created.
booking_customerId	varchar2		☐	Unique identifier of the customer linked to the booking.
booking_discount	decimal		☐	Total discount applied.
booking_externalId	varchar2		☐	External booking ID.
booking_fees	decimal		☐	Total fees applied.
booking_name	varchar2		☐	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
booking_remainder	decimal		☐	Total remaining cost/outstanding balance.
booking_source	varchar2		☐	The application that created the booking.
booking_status	varchar2		☐	Booking status.
booking_total	decimal		☐	Total cost.
booking_uniqueID	guid		☐	Unique booking ID.
bookingDate	datetime	Booking Date	☐	The date this item was purchased or the date this item is valid for.
bookingEndDate	datetime		☐	The date this item is valid until (this can change after first check-in dependend in product configuration).
bookingItemId	int32		☐	Unique identifier for an item.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
endTime	varchar2		☐	Session end time in 24hr format e.g. 13:30 = 1:30 PM.
productId	int32	Product ID	☐	Unique identifier for a product.
quantity	int32	Quantity	☐	Number of items purchased - same as number of tickets except when GroupSize is greater than one.

Name	Data Type	Label	Required	Documentation
startTime	varchar2		☐	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
endDate		☑		Search end date, based on record modified date.
startDate		☑		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
bookingCreatedByStaffId	varchar2		☐	Staff Id that created the Booking.
bookingCreatedDate	datetime		☑	Date the booking w as created.
bookingCustomerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the booking.

Name	Data Type	Label	Required	Documentation
bookingDate	datetime	Booking Date	☑	Booking date of the ticket - when the ticket is valid for/from.
bookingLocation	varchar2	Location	☑	Sales channel the booking was created through.
bookingModifiedDate	datetime	Modified	☑	Date the booking was most recently modified.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
bookingStatus	varchar2	Status	☑	Payment status of the booking.
createdDate	datetime	Created	☑	Date the booking item was created.
discountAmount	decimal		☐	The amount discounted against this specific booking item.
groupSize	int32	Group Size	☑	The number of guests per quantity - eg. GroupSize of 4 with Quantity 3 would be 12 guests.
productId	varchar2	Product ID	☑	Unique identifier of the product.
quantity	int32	Quantity	☑	Quantity of the product on this booking item.
sessionEnd	varchar2	Session End	☐	End time of the item if applicable.
sessionStart	varchar2	Session Start	☐	Start time of the item if applicable.
taxExemptId	varchar2		☐	The Tax ID provided when overriding the tax amount.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.
startDate		☑		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	varchar2		☐	StaffId of the staff member w ho provided their passcode to process a refund.
bookingPaymentId	varchar2	Booking Payment ID	☑	Unique Identifier of a transaction.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
createdDate	datetime	Created	☑	Date of the transaction.
creditCardLast4Digits	varchar2		☐	Last 4 digits of the credit card used to take payment.
deviceId	varchar2		☐	Unique Identifier of the POS/SSK device used to process the transaction.
receiptNumber	varchar2	Receipt Number	☑	
staffId	varchar2	Staff ID	☐	Identifier of the staff member w ho processed the transaction.
tip	decimal	Tip	☐	Gratuity amount against the transaction.
total	decimal	Total	☑	Value of the transaction.
transactionFeeAmount	decimal	Transaction Fee Amount	☐	The value of the transaction fee against a transaction.
transactionId	varchar2	Transaction ID	☐	The unique payment identifier from the payment gatew ay - or gift card number for gift card payments.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
date		☐		Date of the bookings you are looking for.
keywords		☐		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.
locationIds		☐		Comma separated capacity location IDs. Date is required.
productIds		☐		Comma separated product IDs. Date is required.
startTime		☐		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	varchar2	Booking Reference	☑	Unique identifier for a booking.
createdDate	datetime	Created	☑	Date the booking item was created.

Name	Data Type	Label	Required	Documentation
customerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the booking.
name	varchar2	Name	☐	Name of the booking can be different to the booking owner (defaults to the booking owner's name).
status	varchar2		☐	Booking status.
total	decimal	Total	☐	Total cost.
uniqueID	guid		☐	Unique booking ID.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.
startDate		☑		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	varchar2	Booking Reference	☑	Unique identifier for a booking.
createdDate	datetime	Created	☑	Date the BookingSignedWaiver record was created.
modifiedDate	datetime	Modified	☑	Most recent date the BookingSignedWaiver record was modified.
receiptNumber	varchar2	Receipt Number	☑	
signedWaiverId	varchar2	Signed Waiver ID	☑	Unique identifier for the waiver record.
ticketId	varchar2	Ticket ID	☐	Unique identifier for a ticket within a booking.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
customerId		☑		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
acceptMarketing	char	Accept Marketing	☑	Whether or not the customer has accepted.
addressCity	varchar2		☐	City of the customer.
addressCountry	varchar2		☐	Country of the customer.
addressPostcode	varchar2		☐	Postcode of the customer.
addressState	varchar2		☐	State of the customer.
addressStreet	varchar2		☐	Street of the customer.
addressSuburb	varchar2		☐	Suburb of the customer.
dateOfBirth	datetime	Date of Birth	☐	Date of birth of the customer.
email	varchar2	Email	☐	Email address of the customer.
firstName	varchar2	First Name	☐	First name of the customer.
lastName	varchar2	Last Name	☐	Last name of the customer.
phone	varchar2		☐	Phone number of the customer.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.
startDate		☑		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
acceptMarketing	char	Accept Marketing	☑	Whether or not the customer has accepted.
contactNumber	varchar2	Contact Number	☐	Contact phone number of the customer.
country	varchar2	Country	☐	Country of address of the customer.
createdDate	datetime	Created	☑	Date the customer record w as created.
customerId	varchar2	Customer ID	☑	Unique identifier of the customer.
dateOfBirth	datetime	Date of Birth	☐	Date of birth of the customer.
email	varchar2	Email	☐	Email address of the customer.
firstName	varchar2	First Name	☐	First name of the customer.
gender	varchar2	Gender	☐	Gender of the customer.
lastName	varchar2	Last Name	☐	Last name of the customer.
modifiedDate	datetime	Modified	☑	Date the customer record w as most recently modified.
postcode	varchar2	ZIP Code	☐	Postcode of the customer.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
createdDate	datetime	Created	☐	The date the device w as created.

Name	Data Type	Label	Required	Documentation
deviceId	varchar2		☐	Unique identifier of the POS Device.
deviceName	varchar2		☐	Name of the device.
deviceStatus	varchar2		☐	Active = Currently usable device, Inactive = Deleted device.
deviceType	int32		☐	0 = POS device, 1 = SSK device, 2 = Check In Scanner device.

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 UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐	Dollar amount off the product cost.
discount_bookingDateRestrictionDateRangeDays	varchar2		☐	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		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionDteRangeStartDate	datetime		☐	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	varchar2		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionFromType	varchar2		☐	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	varchar2		☐	The usage limits for each code per booking
discount_bookingRuleType	varchar2		☐	1 = Across selected products (quantity purchased counts towards uses).

Name	Data Type	Label	Required	Documentation
discount_codeGenerationMode	varchar2		☑	How discount codes are generated.
discount_discountId	varchar2		☑	Unique identifier of the discount.
discount_endDate	datetime		☐	Last day the discount can be used.
discount_isSingleUseCode	char		☑	When true, a code for this discount can only be used once.
discount_maxApplicableAmount	decimal		☐	Maximum discount that can be applied to a booking using this discount.
discount_name	varchar2		☑	Discount name.
discount_percentOff	decimal		☐	Percentage off the product cost.
discount_startDate	datetime		☐	First day the discount can be used.
discount_usageLimitNumberOfUses	varchar2		☐	The usage limits for each code in total.
discount_usageLimitType	varchar2		☐	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	varchar2	Discount Code	☑	Discount code.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
discount_amountOff	decimal		☐	Dollar amount off the product cost.
discount_bookingDateRestrictionDateRangeDays	varchar2		☐	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		☐	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.
discount_bookingDateRestrictionDteRangeStartDate	datetime		☐	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	varchar2		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
discount_bookingDateRestrictionFromType	varchar2		☐	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	varchar2		☐	The usage limits for each code per booking
discount_bookingRuleType	varchar2		☐	1 = Across selected products (quantity purchased counts towards uses).
discount_codeGenerationMode	varchar2		☑	How discount codes are generated.
discount_discountId	varchar2		☑	Unique identifier of the discount.
discount_endDate	datetime		☐	Last day the discount can be used.
discount_isSingleUseCode	char		☑	When true, a code for this discount can only be used once.
discount_maxApplicableAmount	decimal		☐	Maximum discount that can be applied to a booking using this discount.
discount_name	varchar2		☑	Discount name.
discount_percentOff	decimal		☐	Percentage off the product cost.
discount_startDate	datetime		☐	First day the discount can be used.
discount_usageLimitNumberOfUses	varchar2		☐	The usage limits for each code in total.
discount_usageLimitType	varchar2		☐	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	varchar2	Product ID	☑	Product ID.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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	☐	Dollar amount off the product cost.
bookingDateRestrictionDateRangeDays	varchar2		☐	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		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionDteRangeStartDate	datetime		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionFromNumber	varchar2		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingDateRestrictionFromType	varchar2		☐	Booking dates this discount can be applied on - this is the date the booking is for not the date the booking is created.
bookingRuleNumberOfUses	varchar2		☐	The usage limits for each code per booking
bookingRuleType	varchar2		☐	1 = Across selected products (quantity purchased counts towards uses).
codeGenerationMode	varchar2	Code Generation Mode	☑	How discount codes are generated.
discountId	varchar2	Discount ID	☑	Unique identifier of the discount.
endDate	datetime	End Date	☐	Last day the discount can be used.
isSingleUseCode	char	Is Single Use Code	☑	When true, a code for this discount can only be used once.
maxApplicableAmount	decimal	Maximum Applicable Amount	☐	Maximum discount that can be applied to a booking using this discount.
name	varchar2	Name	☑	Discount name.
percentOff	decimal	Percent Discount	☐	Percentage off the product cost.
startDate	datetime	Start Date	☐	First day the discount can be used.
usageLimitNumberOfUses	varchar2		☐	The usage limits for each code in total.

Name	Data Type	Label	Required	Documentation
usageLimitType	varchar2		☐	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.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
endDate		☑		Search end date, based on record modified date.
startDate		☑		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
balance	decimal	Balance	☑	The current balance of the gift card.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
createdDate	varchar2	Created	☑	Date the gift card was created.
expiryDate	varchar2	Expires	☐	Date the gift card expires.
giftCardId	varchar2	Gift Card ID	☑	Unique identifier of the gift card.
giftCardNumber	varchar2	Gift Card Number	☑	The identifier used to redeem the gift card - unique per venue.
initialValue	decimal	Initial Value	☑	The original balance of the gift card at purchase.
lastUsedDate	varchar2	Last Used	☐	Most recent date the gift card was redeemed.
modifiedDate	varchar2	Modified	☑	Most recent date the gift card was modified.
ticketId	varchar2	Ticket ID	☑	Unique identifier for a ticket within a booking.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
allowMultipleBookings	char	Allow Multiple Bookings	☑	Whether the location allows multiple different bookings to occur within it at the same time.
locationId	varchar2		☑	Unique identifier of the location.
name	varchar2	Name	☐	Location name.
parentLocationId	varchar2	Parent Location ID	☑	Unique identifier of the location group if applicable.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
bookingReferences		☐		Unique identifier for a booking
date		☐		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
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
comment	varchar2		☐	The note provided by the staff member issuing the credit.
createdDate	datetime	Created	☑	Date the w aiver record w as created.
creditValue	decimal		☑	Value of the credit issued.
modifiedDate	datetime	Modified	☑	Date the w aiver record w as most recently modified.
status	varchar2		☐	The status of the membership credit - can be 'new ', 'applied' or 'voided'.
ticketId	varchar2	Ticket ID	☐	Unique identifier for a ticket.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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
bookingReferences		☐		Unique identifier for a booking
date		☐		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
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
redemptionDate	datetime		☑	The date/time of the redemption.
ticketId	varchar2	Ticket ID	☐	Unique identifier for a ticket.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐		Unique identifier for a booking
date		☐		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	varchar2	Booking Reference	☑	Unique identifier for a booking.
customTicketId	varchar2		☐	Unique identifier for a custom ticket.
eventDate	datetime	Event Date	☑	The date/time of the membership event.
nextStatus	varchar2		☐	The membership status on the membership after.
previousStatus	varchar2		☐	The membership status on the membership before.
ticketId	varchar2	Ticket ID	☐	Unique identifier for a ticket.

Modifiers

Catalog: Roller

Schema: Data

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

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		☐	The cost of the modifier (as either a $ amount or %).
modifierGroupId	varchar2		☐	Unique identifier of the modifier group
modifierGroupName	varchar2		☐	Name of the modifier group.
modifierId	varchar2		☐	Unique identifier of the modifier
modifierName	varchar2		☐	Name of the modifier.
modifierType	varchar2		☐	Whether the modifier adds a specific $ amount of a % of the cost of the modified item.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		The date you need availabilty for.
ProductCategory		☐		Finds product/availability for a given product category.

Name	Data Type	Required	Default Value	Documentation
ProductIds		☐		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		☑	Availability date.
onlineSalesOpen	char		☑	Indicator of w hether the product is w ithin it's sale period.
productavailability_description	varchar2(150)		☐	Brief description of product.
productavailability_id	varchar2		☑	Product Id.
productavailability_imageUrl	varchar2		☐	Product image.
productavailability_name	varchar2		☑	Name.
productavailability_type	varchar2		☑	Product type.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		The date you need availabilty for.

Name	Data Type	Required	Default Value	Documentation
ProductCategory		☐		Finds product/availability for a given product category.
ProductIds		☐		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		☐	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.
date	datetime		☑	Availability date.
onlineSalesOpen	char		☑	Indicator of w hether the product is w ithin it's sale period.
productavailability_description	varchar2(150)		☐	Brief description of product.
productavailability_id	varchar2		☑	Product Id.
productavailability_imageUrl	varchar2		☐	Product image.
productavailability_name	varchar2		☑	Name.
productavailability_type	varchar2		☑	Product type.
productId	varchar2	Product ID	☑	Product Id.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		The date you need availabilty for.
ProductCategory		☐		Finds product/availability for a given product category.
ProductIds		☐		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	varchar2(150)		☐	Brief description of product.
id	varchar2		☑	Product Id.
imageUrl	varchar2		☐	Product image.
name	varchar2	Name	☑	Name.
type	varchar2		☑	Product type.

ProductAvailabilityProductsByDate

Catalog: Roller

Schema: Data

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

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		☑		The date you need availabilty for.
ProductCategory		☐		Finds product/availability for a given product category.
ProductIds		☐		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		☐	Product cost.
description	varchar2(150)		☐	Brief description of product.
id	varchar2		☑	Product Id.
imageUrl	varchar2		☐	Product image.
name	varchar2	Name	☐	Name.
productavailability_description	varchar2(150)		☐	Brief description of product.
productavailability_id	varchar2		☑	Product Id.
productavailability_imageUrl	varchar2		☐	Product image.
productavailability_name	varchar2		☑	Name.
productavailability_type	varchar2		☑	Product type.
type	varchar2		☐	Product type.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		The date you need availabilty for.
ProductCategory		☐		Finds product/availability for a given product category.
ProductIds		☐		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		☐	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	varchar2(150)		☐	Brief description of product.
productavailability_id	varchar2		☑	Product Id.
productavailability_imageUrl	varchar2		☐	Product image.
productavailability_name	varchar2		☑	Name.
productavailability_type	varchar2		☑	Product type.
productId	varchar2	Product ID	☑	Product Id.
session_capacityRemaining	int32		☐	Summary of remaining capacity. Null value means unlimited capacity Check Allocations for individual remaining capacity per product (ticket type).
session_date	datetime		☑	Session date.
session_endTime	varchar2		☐	Session end time in 24hr format e.g. 11:30 = 11:30 AM.
session_name	varchar2		☐	Session name.
session_onlineSalesOpen	char		☑	Indicator of w hether the product is w ithin it's sale period.
session_startTime	varchar2		☐	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		The date you need availabilty for.
ProductCategory		☐		Finds product/availability for a given product category.
ProductIds		☐		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		☐	Summary of remaining capacity. Null value means unlimited capacity Check Allocations for individual remaining capacity per product (ticket type).
date	datetime		☑	Session date.
endTime	varchar2		☐	Session end time in 24hr format e.g. 11:30 = 11:30 AM.
name	varchar2	Name	☐	Session name.
onlineSalesOpen	char		☑	Indicator of w hether the product is w ithin it's sale period.
productavailability_description	varchar2(150)		☐	Brief description of product.

Name	Data Type	Label	Required	Documentation
productavailability_id	varchar2		☑	Product Id.
productavailability_imageUrl	varchar2		☐	Product image.
productavailability_name	varchar2		☑	Name.
productavailability_type	varchar2		☑	Product type.
startTime	varchar2		☐	Session start time in 24hr format e.g. 11:30 = 11:30 AM.

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 UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐	Product cost.
hqCode	varchar2		☐	Unique identifier of the HQ product it is linked to.
hqProductId	varchar2		☐	Unique identifier of the HQ parent product it is linked to.
name	varchar2	Name	☐	Product name.
parentProductId	varchar2	Parent Product ID	☐	Unique identifier of the parent product.
productId	varchar2	Product ID	☑	Unique identifier of the product.
productStatus	varchar2		☑	Published status of the product.
productSubType	varchar2	Product Sub-type	☐	The sub type/variation of the product.
productType	varchar2	Product Type	☑	The type/variation of the product.
reportingCategoryName	varchar2	Reporting Category Name	☐	Reporting category name the product is allocated to.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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	varchar2	General Ledger Account Code	☐	General ledger (GL) code.
name	varchar2	Name	☑	Reporting Category name.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
glCode	varchar2	General Ledger Account Code	☐	General ledger (GL) code.
name	varchar2	Name	☑	Reporting Category name.
parent_glCode	varchar2		☐	Parent general ledger (GL) code.
parent_name	varchar2		☑	Parent reporting Category name.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
category_glCode	varchar2		☐	General ledger (GL) code.
category_name	varchar2		☑	Reporting Category name.
productId	varchar2	Product ID	☑	Product ID.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☐		Booking references specified as list in comma-separated format. Maximum limit is 100.
endDate		☐		Search end date, based on record modified date.
startDate		☐		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	☑	The accounts receivable generated or removed for this entry.
bookingPaymentId	varchar2	Booking Payment ID	☐	Unique Identifier of a transaction.
bookingReference	varchar2	Booking Reference	☑	Unique identifier for a booking.
customerId	varchar2	Customer ID	☐	Unique identifier of the customer for the booking.
deferredRevenue	decimal	Deferred Revenue	☐	The deferred revenue generated or removed for this entry.
deferredRevenueGiftCards	decimal		☐	The giftcard deferred revenue generated or removed for this entry.
deferredRevenueOther	decimal		☐	The external deferred revenue generated or removed for this entry (used for integrations, eg. Groupon).
discount	decimal	Discount	☐	The value of the discount against a transaction entry.
discountCode	varchar2	Discount Code	☐	The discount code used.
eventDate	datetime	Event Date	☑	The date the entry occurred - not necessarily the date the entry w as generated, but w hen the values should be recorded as occurring.
eventType	varchar2	Event Type	☑	The type of entry - Transaction, Redemption or Expiry.
externalPaymentReference	varchar2	External Payment Reference	☐	The unique payment identifier from the payment gatew ay - or gift card number for gift card payments.
feeRevenue	decimal	Fee Revenue	☑	The amount of revenue from transaction fees for this entry.
fundsReceived	decimal	Funds Received	☑	The amount of money received for a transaction entry.
netRevenue	decimal	Net Revenue	☑	The after tax revenue generated for this entry.
paymentType	varchar2	Payment Type	☑	The type of tender used for the transaction.
productId	varchar2	Product ID	☑	Unique identifier of a product.
productType	varchar2	Product Type	☑	Passes, Add-ons, Custom Products, Gift Cards, Package, Wallet.
receiptNumber	varchar2	Receipt Number	☑	
recognisedDiscount	decimal	Recognized Discount	☐	The value of the discount against a redemption or expiry entry.
taxOnFundsReceived	decimal	Tax on Funds Received	☐	The amount of tax on funds received for a transaction entry.
taxPayable	decimal	Tax Payable	☑	The amount of tax payable on recognised revenue for this entry.

Name	Data Type	Label	Required	Documentation
taxPercent	decimal	Tax Percentage	☑	The tax percent against the product in this entry.
ticketId	varchar2	Ticket ID	☑	Unique identifier for a ticket within a booking.
ticketTransactionValue	decimal	Ticket Transaction Value	☑	The value of the amount paid against the booking that is being attributed to this entry.
ticketUnitCost	decimal	Ticket Unit Cost	☑	The price of the ticket being referenced in the entry.
transactionDate	datetime	Transaction Date	☑	Date of the first transaction against a booking.
transactionFeeAmount	decimal	Transaction Fee Amount	☐	The value of the transaction fee against a transaction entry.
transactionLocation	varchar2	Transaction Location	☑	The sales channel the booking was generated from.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.

Name	Data Type	Required	Default Value	Documentation
startDate		☑		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	varchar2	Contact Number	☐	Contact phone number of the waiver holder. Not returned for minors.
createdDate	datetime	Created	☑	Date the waiver record was created.
customerId	varchar2	Customer ID	☐	Unique identifier of the customer record of the waiver holder.
dateOfBirth	datetime	Date of Birth	☐	Date of birth of the waiver holder.
email	varchar2	Email	☐	Email address of the waiver holder. Not returned for minors.
expiryDate	datetime	Expires	☑	Date the waiver expired.
firstName	varchar2	First Name	☐	First name of the customer as completed on the waiver.
isForMinor	char	Is for Minor	☑	Identifies whether the customer is a minor or not at the time of signing. Returns true for minors, false for adults.
isValid	char	Is Valid	☐	Only returned if waiver verification is required. Returns true for verified waivers and false for unverified.
lastName	varchar2	Last Name	☐	Second name of the customer as completed on the waiver.
modifiedDate	datetime	Modified	☑	Date the waiver record was most recently modified.
parentSignedWaiverId	varchar2	Parent Signed Waiver ID	☐	Only returned for minor's waivers - will be the SignedWaiverId of their parent/guardians waiver.
signedWaiverId	varchar2	Signed Waiver ID	☑	Unique identifier for the waiver record.
waiverId	varchar2		☑	The identifier of the version of the waiver they have signed.

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 UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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	varchar2	Contact Number	☐	Contact phone number of the staff member.
createdDate	datetime	Created	☐	Date the staff record w as created.
email	varchar2	Email	☐	Email address of the staff member.
firstName	varchar2	First Name	☐	First name of the staff member.
lastName	varchar2	Last Name	☐	Last name of the staff member.
role	varchar2		☐	Staff role / permission level.
staffId	varchar2	Staff ID	☑	Unique identifier for a staff member.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.
startDate		☑		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
discountId	varchar2	Discount ID	☑	Unique identifier of the discount valid for this ticket.
ticket_bookingDate	datetime		☐	Booking date of the ticket - when the ticket is valid for/from.
ticket_createdDate	datetime		☑	Date the ticket was created.
ticket_customerId	varchar2		☐	Unique identifier of the customer linked to the ticket.
ticket_customTicketId	varchar2		☐	Ticket identifier specified manually by the user e.g. the code from a pre-printed membership card.
ticket_expiryDate	datetime		☑	Date the ticket is no longer valid.
ticket_name	varchar2		☐	Ticket holder name.
ticket_numberOfRecurringPayments	int32		☐	Number of recurring payments.
ticket_productId	varchar2		☐	Unique identifier of the product the ticket is for.
ticket_productSubType	varchar2		☐	The sub type/variation of the product.
ticket_productType	varchar2		☑	The type/variation of the product.
ticket_recurringPaymentFrequency	varchar2		☐	Recurring payment frequency.
ticket_ticketId	varchar2		☑	Unique identifier of the ticket.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

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		☑		Search end date, based on record modified date.
startDate		☑		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	☐	Booking date of the ticket - when the ticket is valid for/from.
createdDate	datetime	Created	☑	Date the ticket was created.
customerId	varchar2	Customer ID	☐	Unique identifier of the customer linked to the ticket.
customTicketId	varchar2		☐	Ticket identifier specified manually by the user e.g. the code from a pre-printed membership card.
expiryDate	datetime	Expires	☑	Date the ticket is no longer valid.
name	varchar2	Name	☐	Ticket holder name.
numberOfRecurringPayments	int32	Number of Recurring Payments	☐	Number of recurring payments.
productId	varchar2	Product ID	☐	Unique identifier of the product the ticket is for.
productSubType	varchar2	Product Sub-type	☐	The sub type/variation of the product.
productType	varchar2	Product Type	☑	The type/variation of the product.

Name	Data Type	Label	Required	Documentation
recurringPaymentFrequency	varchar2	Recurring Payment Frequency	☐	Recurring payment frequency.
ticketId	varchar2	Ticket ID	☑	Unique identifier of the ticket.

TillReconciliations

Catalog: Roller

Schema: Data

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

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		☑		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		☐	The amount of cash that was counted and reconciled as being in the till at the end of the till session.
cardRefunds	decimal		☐	Credit Card refunds for this till session.
cardTakings	decimal		☐	Credit Card takings for this till session.
cashAdded	decimal		☐	The amount of cash that was manually added to the till during the till session.

Name	Data Type	Label	Required	Documentation
cashFloat	decimal		☐	The amount of cash that was in the till at the beginning of the till session.
cashRefunds	decimal		☐	Cash Refunds for this till session.
cashRemoved	decimal		☐	The amount of cash that was manually removed from the till during the till session.
cashTakings	decimal		☐	Cash takings for this till session.
cashVariance	decimal		☐	The difference between ExpectedCashBalance and ActualCashBalance. A positive number indicates excess cash a negative number indicates a shortage.
checkTakings	decimal		☐	Check takings for this till session.
deviceId	varchar2		☐	Unique identifier of the POS Device.
endDateDate	datetime		☐	Date/time the till session was closed and reconciled.
expectedCashBalance	decimal		☐	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		☐	Gift Card takings for this till session (transactions paid for by gift cards).
startDate	datetime	Start Date	☐	Date/time of the first transaction for this till session.

Venues

Catalog: Roller

Schema: Data

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

Table Columns

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

Name	Data Type	Label	Required	Documentation
codeOfConduct	varchar2		☐	The terms and conditions set by the venue for guests when they make a purchase.
country	varchar2	Country	☐	Country where the venue is located.
countryCode	varchar2		☐	Country code where the venue is located.

Name	Data Type	Label	Required	Documentation
creditCardFeePercent	decimal		☐	The % amount of fees charged on Card payments on the Online Checkout and in Venue Manager.
currency	varchar2		☐	Trading currency.
feePerTransaction	decimal		☐	The $ amount of fees charged to guests for Online Checkout payments.
fees	decimal		☐	Amount of booking fees applied to the booking.
id	int32		☐	Roller Venue ID.
locale	varchar2		☐	Locale code where the venue is located.
name	varchar2	Name	☐	Name of the venue.
paymentSettingsApiUrl	varchar2		☐	URL for the payment service.
paymentSettingsConfigurationId	varchar2		☐	
paymentSettingsIntegrationId	varchar2		☐	
productPricesIncludeTax	char		☐	Whether or not the prices of products for the venue include taxes.
timeZone	varchar2		☐	Local time zone.
transactionFeePercent	decimal		☐	The % amount of fees charged to guests for Online Checkout payments.

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 Invantive UniversalSQL driver for Roller does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Roller API.

Table Columns

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

Name	Data Type	Label	Required	Documentation
createdDate	datetime	Created	☑	Date the waiver was created.
minorAgeLimitsInYears	int32	Minor Age Limits in Years	☑	What age a customer must be to be considered an adult when signing the waiver.
name	varchar2	Name	☑	Name of the waiver.
requiresSignature	char	Requires Signature	☑	Whether or not the waiver requires a signature.
terms	varchar2	Terms	☑	Terms of the waiver.

Name	Data Type	Label	Required	Documentation
validityInDays	int32	Validity in Days	☑	How long the waiver is valid for once signed.
waiverId	varchar2		☑	Unique identifier of the waiver the customers are signing.

Webhooks

Catalog: Roller

Schema: Data

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

Table Columns

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

Name	Data Type	Label	Required	Documentation
createdDate	datetime	Created	☐	Date the staff record was created.
webhookId	int32		☑	Webhook ID.

3.1.2 Schema: Native

3.1.2.1 Tables

NATIVEPLATFORMSCALARREQUESTS: Roller Native Platform Scalar Requests

{res:itgen_native_platform_scalar_requests_desc}

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	char	BLOB Preferred	☑	Indicator w hether a BLOB result is preferred over text.
BOL_RESPONSE_CACHE_MAX_AGE_SEC	int32	Response Cache Maximum Age (sec)	☐	Maximum age in seconds of Bridge Online response cache entries to be used.
CONTENT_TYPE	varchar2(240)	Content Type	☐	
DATE_ENDED	datetime	End Date	☑	
DATE_STARTED	datetime	Start Date	☑	
DRY_RUN	char	Run w ithout Actions	☑	
DURATION_MS	int64	Duration (ms)	☑	
ERROR_MESSAGE_CODE	varchar2(30)	Error Message Code	☐	
ERROR_MESSAGE_TEXT	varchar2(32000)	Error Message Text	☐	
FAIL_ON_ERROR	char	Fail on Error	☑	Whether to raise an exception w hen processing the native request triggered an error from the provider.
HTTP_DISK_CACHE_MAX_AGE_SEC	int32	HTTP Disk Cache Maximum Age (sec)	☐	Maximum age in seconds of HTTP disk cache entries to be used.
HTTP_DISK_CACHE_SAVE	char	Save HTTP Disk Cache	☐	Whether results can be stored in HTTP disk cache.
HTTP_DISK_CACHE_USE	char	Use HTTP Disk Cache	☐	Whether results can be fetched from HTTP disk cache.
HTTP_MEMORY_CACHE_MAX_AGE_SEC	int32	HTTP Memory Cache Maximum Age (sec)	☐	Maximum age in seconds of HTTP memory cache entries to be used.
HTTP_MEMORY_CACHE_SAVE	char	Save HTTP Memory Cache	☐	Whether results can be stored in HTTP memory cache.
HTTP_MEMORY_CACHE_USE	char	Use HTTP Memory Cache	☐	Whether results can be fetched from HTTP memory cache.
HTTP_METHOD	varchar2(30)	HTTP Method	☐	
HTTP_STATUS_CODE	numeric	HTTP Status Code	☐	
ORIG_SYSTEM_GROUP	varchar2(4000)	Original System Group	☐	
ORIG_SYSTEM_REFERENCE	varchar2(4000)	Original System Reference	☐	
PAYLOAD_TEXT	varchar2	Payload	☐	
RESULT_BLOB	blob	Result BLOB	☐	
RESULT_DATE_TIME_UTC	datetime	Result Date Time	☐	
RESULT_NUMBER	number	Result Number	☐	
RESULT_TEXT	varchar2	Result Text	☐	
SUCCESSFUL	char	Succesful	☑	
TIMEOUT_SEC	int32	Timeout (sec)	☐	Timeout in seconds.

Name	Data Type	Label	Required	Documentation
TRANSACTION_ID	int64	Transaction ID	☑	Incrementing ID of the transaction.
URL	varchar2(4000)	URL	☐	

4 Package: dcr_metadata

4.1 Procedures

4.1.1 dcr_metadata.get_partitions: Roller Data container metadata package

Get all partitions.

Documentation:

List all partitions.

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- Z -

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

24.0.92
28/05/2026, 19:54
ISBN 978-94-6302-000-8: