# Tixly API Data Model

*for use with Invantive SQL*

23.0

## Important Safety and Usage Information

Intended Use and Limitations: This softw are, 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 softw are is not designed, tested, or approved for use in environments w here 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 softw are should not be used for operations or functions related to the control, maintenance, or operation of nuclear facilities.
- Defense and Military Applications: This softw are is not suitable for use in defense-related applications, including but not limited to w eaponry control, military strategy planning, or any other aspects of national defense.
- Aviation: The softw are 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 softw are should not be utilized for medical device operation, patient data analysis for critical health decisions, pharmaceutical production, or medical research w here its failure or malfunction could impact patient health.
- Chemical and Hazardous Material Handling: This softw are is not intended for the management, control, or operational aspects of chemical plants or hazardous material handling facilities. Any malfunction in softw are used in these settings could result in dangerous chemical spills, explosions, or environmental disasters.
- Transportation and Traffic Control Systems: The softw are should not be used for the control, operation, or management of transportation systems, including railw ay signal controls, subw ay 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 softw are is not designed for the control or operation of energy grid systems, including electrical substations, renew able energy control systems, or w ater utility control systems. The failure of softw are in these areas could lead to significant pow er outages, w ater supply disruptions, or other public utility failures, potentially endangering communities and causing extensive damage.
- Other High-Risk Environments: Any other critical infrastructure and environments w here a failure of the softw are could result in significant harm to individuals or the environment.

User Responsibility: Users must ensure that they understand the intended use of the softw are 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 softw are 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 softw are in prohibited or unintended applications.

# Contents

# 1 SQL Driver for Tixly API

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

Tixly is online software for leisure and entertainment management. Tixly is available globally.

The Tixly driver covers 79 tables and 1847 columns.

## Tixly API Clients

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

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

The Tixly Power BI connector is based on the Invantive SQL driver for Tixly, 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 Tixly 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 Tixly 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 Tixly ADO.net provider.

Finally, online web apps can be build for Tixly using App Online of Invantive Cloud.

## Monitor API Calls

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

## Specifications

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

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

The configuration can be changed using various attributes 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 Tixly the comparison of two texts is case sensitive by default.

Changes and bug fixes on the Tixly SQL driver can be found in the release notes. Get access to the community through the Tixly section of the Invantive forums.

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

Alias: `tixly`

Recommended alias: `tly`

Driver code for use in settings.xml

Updated 22-12-2022 21:15 using Invantive SQL version 22.1.101-BETA+3681.

# 2 SQL Driver Attributes for Tixly API

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

The Tixly 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 Tixly 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 Tixly 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 row s to be unique for softw are analysis. | Shared | False | ✓ | ✓ | ✓ | |
| api-access-token | Access Token is a security token for multiple OAuth2 Flow s. With an Access Token you can access protected resources. An Access Token must be stored securely since once compromised allow s 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 passw ord 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 w ith 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 w ebsite a brow ser 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 renew ed access token to access protected resources. A Refresh Token and client secret must be stored securely since once compromised allow s access to your protected resources. | OData | | ✓ | | ✓ | ✓ |
| 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 w ith. | OData | | ✓ | | ✓ | |
| api-url | URL of w eb service. | OData | | ✓ | | ✓ | |
| bulk-delete-page-size-row s | Number of row s to delete per batch w hen bulk deleting. | Shared | 10000 | ✓ | ✓ | ✓ | |
| bulk-insert-page-size-bytes | Approximate maximum size in bytes of batch w hen bulk inserting. | Shared | 10000000 | ✓ | ✓ | ✓ | |
| bulk-insert-page-size-row s | Number of row s to insert per batch w hen bulk inserting. | Shared | 250 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| dow nload-error-400-bad-request-max-tries | Maximum number of tries w hen HTTP server reports bad format during retrieval of data. | | 3 | ✓ | ✓ | ✓ | |
| dow nload-error-400-bad-request-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen HTTP server reports that the API server is unavailable during retrieval of data. | | 500 | ✓ | ✓ | ✓ | |
| dow nload-error-400-bad-request-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen HTTP server reports that the API server is unavailable during retrieval of data. | | 5000 | ✓ | ✓ | ✓ | |
| dow nload-error-400-bad-request-sleep-multiplicator | Multiplication factor for sleep betw een retries HTTP server reports that the API server is unavailable during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-408-request-timeout-max-tries | Maximum number of tries w hen the w ebsite reports a HTTP status 408. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-408-request-timeout-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the w ebsite reports a HTTP status 408. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-408-request-timeout-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the w ebsite reports a HTTP status 408. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-408-request-timeout-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the w ebsite reports a HTTP status 408. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-422-bad-request-max-tries | Maximum number of tries w hen HTTP server reports unprocessable entity during retrieval of data. | | 30 | ✓ | ✓ | ✓ | |
| dow nload-error-422-bad-request-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen HTTP server reports unprocessable entity during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-422-bad-request-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen HTTP server reports unprocessable entity during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-422-bad-request-sleep-multiplicator | Multiplication factor for sleep betw een retries HTTP server reports unprocessable entity during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-429-too-many-requests-max-tries | Maximum number of tries w hen the w ebsite reports that too many requests have been made during a timeslot of one minute or one day. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-429-too-many-requests-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the w ebsite reports that too many requests have been made during a timeslot of one minute or one day. | | 10000 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| dow nload-error-429-too-many-requests-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the w ebsite reports that too many requests have been made during a timeslot of one minute or one day. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-429-too-many-requests-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the w ebsite reports that too many requests have been made during a timeslot of one minute or one day. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-502-server-unavailable-max-tries | Maximum number of tries w hen HTTP server reports a bad gatew ay during retrieval of data. | | 30 | ✓ | ✓ | ✓ | |
| dow nload-error-502-server-unavailable-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen HTTP server reports a bad gatew ay during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-502-server-unavailable-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen HTTP server reports that a bad gatew ay during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-502-server-unavailable-sleep-multiplicator | Multiplication factor for sleep betw een retries HTTP server reports a bad gatew ay during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-503-server-unavailable-max-tries | Maximum number of tries w hen HTTP server reports that the API server is unavailable during retrieval of data. | | 30 | ✓ | ✓ | ✓ | |
| dow nload-error-503-server-unavailable-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen HTTP server reports that the API server is unavailable during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-503-server-unavailable-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen HTTP server reports that the API server is unavailable during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-503-server-unavailable-sleep-multiplicator | Multiplication factor for sleep betw een retries HTTP server reports that the API server is unavailable during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-504-gatew ay-timeout-max-tries | Maximum number of tries w hen the w ebsite reports a gatew ay timeout. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-504-gatew ay-timeout-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the w ebsite reports a gatew ay timeout. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-504-gatew ay- | Maximum sleep in milliseconds betw een retries w hen the w ebsite reports a gatew ay timeout. | | 300000 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| timeout-sleep-max-ms | | | | | | | |
| 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 | ✓ | ✓ | ✓ | |
| 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- | Multiplication factor for sleep between retries when an argument | | 2 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| sleep-multiplicator | exception is returned w hen dow nloading a blob. | | | | | | |
| dow nload-error-internet-dow n-max-tries | Maximum number of tries w hen the Internet connection seems dow n during retrieval of data. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-internet-dow n-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the Internet connection seems dow n during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-internet-dow n-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the Internet connection seems dow n during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-internet-dow n-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the Internet connection seems dow n during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-io-exception-max-tries | Maximum number of tries w hen a netw ork I/O connection failure occurs during retrieval of data. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-io-exception-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen a netw ork I/O connection failure occurs during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-io-exception-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen a netw ork I/O connection failure occurs during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-io-exception-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen a netw ork I/O connection failure occurs during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-json-exception-max-tries | Maximum number of tries w hen an invalid JSON body is returned. | | 3 | ✓ | ✓ | ✓ | |
| dow nload-error-json-exception-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen an invalid JSON body is returned. | | 1000 | ✓ | ✓ | ✓ | |
| dow nload-error-json-exception-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen an invalid JSON body is returned. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-json-exception-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen an invalid JSON body is returned. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-other-exception-max-tries | Maximum number of tries w hen an unqualified error occurs during retrieval of data. | | 3 | ✓ | ✓ | ✓ | |
| dow nload-error-other-exception-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen an unqualified error ocurrs during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| dow nload-error-other-exception-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen an unqualified error ocurrs during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-other-exception-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen an unqualified error ocurrs during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-socket-exception-max-tries | Maximum number of tries w hen the netw ork connection is forcible dropped during retrieval of data. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-socket-exception-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the netw ork connection is forcible dropped during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-socket-exception-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the netw ork connection is forcible dropped during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-socket-exception-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the netw ork connection is forcible dropped during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-exception-max-tries | Maximum number of tries w hen a w eb connection failure occurs during retrieval of data. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-exception-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen a w eb connection failure occurs during retrieval of data. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-exception-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen a w eb connection failure occurs during retrieval of data. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-exception-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen a w eb connection failure occurs during retrieval of data. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-not-implemented-max-tries | Maximum number of tries w hen the connection reports not implemented. | | 1 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-not-implemented-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the connection reports not implemented. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-not-implemented-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the connection reports not implemented. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-not- | Multiplication factor for sleep betw een retries w hen the connection reports not implemented. | | 2 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| implemented-sleep-multiplicator | | | | | | | |
| dow nload-error-w eb-timeout-max-tries | Maximum number of tries w hen the connection reports a timeout. | | 10 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-timeout-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the connection reports a timeout. | | 1000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-timeout-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the connection reports a timeout. | | 30000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-timeout-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the connection reports a timeout. | | 2 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-unauthorized-max-tries | Maximum number of tries w hen the connection reports an unauthorized error. | | 1 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-unauthorized-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the connection reports an unauthorized error. | | 10000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-unauthorized-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the connection reports an unauthorized error. | | 300000 | ✓ | ✓ | ✓ | |
| dow nload-error-w eb-unauthorized-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the connection reports an unauthorized error. | | 2 | ✓ | ✓ | ✓ | |
| fail-on-duplicate-row s | When true, an error is raised w hen there are multiple row s w ith identical values in a single table.. | | False | ✓ | ✓ | ✓ | ✓ |
| 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, Low er, Upper and Mixed. | Shared | | ✓ | ✓ | ✓ | |
| http-disk-cache-compression-level | Compression level for the HTTP disk cache, ranging from 1 (little) to 9 (intense). Default is 5. | Shared | 5 | ✓ | ✓ | ✓ | |
| http-disk-cache-directory | Directory w here HTTP cache is stored. | Shared | C:\Users\gle3.WS212\Invantive\Cache\http\gle3\shared | ✓ | ✓ | ✓ | |
| http-disk-cache-ignore-w rite-errors | Whether to ignore w rite errors to disk cache. | Shared | False | ✓ | ✓ | ✓ | |
| http-disk-cache-max-age-sec | Maximum acceptable age in seconds for use of data in the HTTP disk cache. | Shared | 2592000 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| 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. | OData | 5 | ✓ | ✓ | ✓ | |
| http-memory-cache-max-age-sec | Maximum acceptable age in seconds for use of data in the HTTP memory cache. | OData | 14400 | ✓ | ✓ | ✓ | |
| http-post-timeout-max-ms | HTTP POST maximum timeout on retry (ms). | | 58000 | ✓ | ✓ | ✓ | |
| http-post-timeout-ms | HTTP POST timeout (ms). | | 57000 | ✓ | ✓ | ✓ | |
| ignore-http-400-errors | Ignore HTTP 400 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-401-errors | Ignore HTTP 401 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-402-errors | Ignore HTTP 402 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-403-errors | Ignore HTTP 403 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-404-errors | Ignore HTTP 404 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-422-errors | Ignore HTTP 422 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-429-errors | Ignore HTTP 429 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-500-errors | Ignore HTTP 500 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-502-errors | Ignore HTTP 502 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-http-503-errors | Ignore HTTP 503 errors w hen exchanging results w ith the HTTP endpoint. | | False | ✓ | ✓ | ✓ | |
| ignore-unknow n-path-type | Whether to ignore path types not yet supported. An error w ill be generated w hen an unsupported type occurs. | | True | ✓ | ✓ | ✓ | ✓ |
| ignore-values-unknow n-path | Whether to ignore values outside of processed paths. An error w ill be | | True | ✓ | ✓ | ✓ | ✓ |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| | generated w hen a value occurs outside a path otherw ise. | | | | | | |
| invalid-json-on-get-max-tries | Maximum number of tries w hen the JSON received on GET is invalid. | | 1 | ✓ | ✓ | ✓ | |
| invalid-json-on-get-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the JSON received on GET is invalid. | | 1000 | ✓ | ✓ | ✓ | |
| invalid-json-on-get-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the JSON received on GET is invalid. | | 10000 | ✓ | ✓ | ✓ | |
| invalid-json-on-get-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen the JSON received on GET is invalid. | | 2 | ✓ | ✓ | ✓ | |
| invalid-json-on-post-max-tries | Maximum number of tries w hen the JSON received on POST is invalid. | | 1 | ✓ | ✓ | ✓ | |
| invalid-json-on-post-sleep-initial-ms | Initial sleep in milliseconds betw een retries w hen the JSON received on POST is invalid. | | 1000 | ✓ | ✓ | ✓ | |
| invalid-json-on-post-sleep-max-ms | Maximum sleep in milliseconds betw een retries w hen the JSON received on POST is invalid. | | 10000 | ✓ | ✓ | ✓ | |
| invalid-json-on-post-sleep-multiplicator | Multiplication factor for sleep betw een retries w hen 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. Otherw ise they are replaced by 01-01-1753. | SQL Engine V1 | False | ✓ | ✓ | ✓ | |
| invantive-sql-forw ard-filters-to-data-containers | Whether to forw ard 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 w hen executing a join set. | OData | 60 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| limit-partition-calls-left | Minimum number of remaining API calls on a partition towards a hard limit. When below, an error is raised. | OData | 500 | ✓ | ✓ | ✓ | |
| log-native-calls-to-disk-max-events | Maximum number of 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-discovered-column-count | Maximum number of discovered columns. An error will be generated when the column exceeds this value. | | 250 | ✓ | ✓ | ✓ | ✓ |
| 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-url-length-accepted | The maximum accepted URL length before raising an error. | Shared | 8000 | ✓ | ✓ | ✓ | |
| max-url-length-desired | The maximum desired URL length. | Shared | 8000 | ✓ | ✓ | ✓ | |
| metadata-cache-max-age-sec | Maximum acceptable age in seconds for re-use of metadata. | OData | | ✓ | ✓ | ✓ | |
| oauth-unauthorized-max-tries | Maximum number of tries when an OAuth exception occurs. | OData | 2 | ✓ | ✓ | ✓ | |
| oauth-unauthorized-sleep-initial-ms | Initial sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails. | OData | 10000 | ✓ | ✓ | ✓ | |
| oauth-unauthorized-sleep-max-ms | Maximum sleep in milliseconds between OAuth reauthentication tries when the OAuth authentication fails. | OData | 1000 | ✓ | ✓ | ✓ | |
| oauth-unauthorized-sleep-multiplicator | Multiplication factor for sleep between OAuth reauthentication tries when the OAuth authentication fails. | OData | 2 | ✓ | ✓ | ✓ | |
| partition-slot-based-rate-limit-length-ms | Total length in milliseconds across all slots of a partition-based rate limit. | Shared | 60000 | ✓ | | ✓ | |
| partition-slot-based-rate-limit-slots | Number of slots per partition-based rate limit. Null means no slot-based rate limit. | Shared | | ✓ | | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| pre-request-delay-ms | Pre-request delay in milliseconds per request. | Shared | 0 | ✓ | ✓ | ✓ | |
| requested-page-size | Preferred number of rows to exchange per round trip; only 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 | ✓ | ✓ | ✓ | |
| 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 | | 0 | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| | with the HTTP endpoint. | | | | | | |
| 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-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 | ✓ | ✓ | ✓ | |
| 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 | ✓ | ✓ | ✓ | |
| swagger-specification-file | The Swagger file path, such as C:\temp\swagger.json. | | | ✓ | ✓ | ✓ | ✓ |
| swagger-specification-http-disk-cache-max-age-sec | Maximum acceptable age in seconds for use of Swagger specification data in the HTTP disk cache. | | 86400 | ✓ | ✓ | ✓ | |
| swagger-specification-url | The Swagger URL such as https://example.org/rest/swagger.json. | | | ✓ | ✓ | ✓ | ✓ |
| use-batch-insert | Whether to use batch insert. | OData | True | ✓ | ✓ | ✓ | |
| use-http-disk-cache-read | Whether to use HTTP responses from previous queries stored on disk to answer the current query. | Shared | False | ✓ | ✓ | ✓ | |
| use-http-disk-cache-write | Whether to memorize HTTP responses on disk. | Shared | False | ✓ | ✓ | ✓ | |

| Code | Description | Origin | Default Value | Set from Connection String | Set from Set SQL-Statement | Set from Drivers File | Set from Log On |
|---|---|---|---|---|---|---|---|
| use-http-memory-cache-read | Whether to use HTTP responses from previous queries stored in memory that can answer the current query. | OData | True | ✓ | ✓ | ✓ | |
| use-http-memory-cache-write | Whether to memorize HTTP responses from previous queries for use by future queries. | OData | True | ✓ | ✓ | ✓ | |

# 3 Schema: Customer

## 3.1 Tables

### 3.1.1 AnonymizedCustomers: Tixly Anonymized Customers

Gets merged customers details.

Catalog: Tixly

Schema: Customer

Label: Anonymized Customers

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

#### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function AnonymizedCustomers. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | |
| DateTo | datetime | ☐ | | |

#### Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Anonymized | datetime | Anonymized | ☐ | |
| AnonymizedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |

### 3.1.2 CustomerByID: Tixly Customer by ID

Fetches a specific customer by his customer ID

Catalog: Tixly

Schema: Customer

Label: Customer by ID

This is a read-only table function. The Tixly API may not support changing the data or the Invantive SQL driver for Tixly does not cover it. In the latter case, please use the table NativePlatformScalarRequests to upload data to the Tixly 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## 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 |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.3 CustomerDonationSalesByCustomerId: Tixly Customer Donation Sales by Customer ID

Fetches the donations from a specific customer

Catalog: Tixly

Schema: Customer

Label: Customer Donation Sales by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerDonationSalesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Vat | double | VAT | ☐ | |

### 3.1.4 CustomerEventSalesAllocationsByCustomerId: Tixly Customer Event Sales Allocations by Customer ID

Fetches the events that a specific customer has bought a ticket to

Catalog: Tixly

Schema: Customer

Label: Customer Event Sales Allocations by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerEventSalesAllocationsByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Block | boolean | Block | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| ExpiresHoursBefore | int32 | Expires Hours Before | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| ReservedTickets | int32 | Reserved Tickets | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Seats | int32 | Seats | ☐ | |
| Sold | int32 | Sold | ☐ | |
| SoldTickets | int32 | Sold Tickets | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 3.1.5 CustomerEventSalesByCustomerId: Tixly Customer Event Sales by Customer ID

Fetches the events that a specific customer has bought a ticket to

Catalog: Tixly

Schema: Customer

Label: Customer Event Sales by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerEventSalesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllow ed | boolean | Online Reservation Allow ed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 3.1.6 CustomerEventSalesCategoriesByCustomerId: Tixly Customer Event Sales Categories by Customer ID

Fetches the events that a specific customer has bought a ticket to

Catalog: Tixly

Schema: Customer

Label: Customer Event Sales Categories by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerEventSalesCategoriesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 3.1.7 CustomerEventSalesTagsByCustomerId: Tixly Customer Event Sales Tags by Customer ID

Fetches the events that a specific customer has bought a ticket to

Catalog: Tixly

Schema: Customer

Label: Customer Event Sales Tags by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerEventSalesTagsByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 3.1.8 CustomerGiftCardsById: Tixly Customer Gift Cards by ID

Fetches the giftcars that a specific customer has bought

Catalog: Tixly

Schema: Customer

Label: Customer Gift Cards by ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerGiftCardsById. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Amount | double | Amount | ☐ | |
| AmountLeft | double | Amount Left | ☐ | |
| AmountUsed | double | Amount Used | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Expires | datetime | Expires | ☐ | |
| ExpiresUTCUnix | int64 | | ☐ | |
| GiftCardCount | int32 | Giftcard Count | ☐ | |
| Number | string | Number | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Pin | string | PIN | ☐ | |
| Price | double | Price | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |

### 3.1.9 CustomerMembershipSalesByCustomerId: Tixly Customer Membership Sales by Customer ID

Fetches the membership a specific customer has bought

Catalog: Tixly

Schema: Customer

Label: Customer Membership Sales by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerMembershipSalesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

### 3.1.10 CustomerMetadataByID: Tixly Customer Metadata by ID

Fetches a specific customer by his customer ID

Catalog: Tixly

Schema: Customer

Label: Customer Metadata by ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerMetadataByID. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.11 CustomerPlusMetadata: Tixly Customer plus Metadata

Fetches all customers you have access to. Pass in the DateFrom and DateTo query parameters to get customers that have been edited or created in a specific date range.

Catalog: Tixly

Schema: Customer

Label: Customer plus Metadata

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerPlusMetadata. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Created/Edited customer from |
| DateTo | datetime | ☐ | | Created/Edited customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.12 CustomerPlusTags: Tixly Customer plus Tags

Fetches all customers you have access to. Pass in the DateFrom and DateTo query parameters to get customers that have been edited or created in a specific date range.

Catalog: Tixly

Schema: Customer

Label: Customer plus Tags

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerPlusTags. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Created/Edited customer from |
| DateTo | datetime | ☐ | | Created/Edited customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.13 CustomerProductSalesByCustomerId: Tixly Customer Product Sales by Customer ID

Fetches the products a specific customer has bought

Catalog: Tixly

Schema: Customer

Label: Customer Product Sales by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerProductSalesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| ProductCount | int32 | Product Count | ☐ | |
| ProductId | int32 | Product ID | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |

### 3.1.14 Customers: Tixly Customers

Fetches all customers you have access to. Pass in the DateFrom and DateTo query parameters to get customers that have been edited or created in a specific date range.

Catalog: Tixly

Schema: Customer

Label: Customers

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Customers. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Created/Edited customer from |
| DateTo | datetime | ☐ | | Created/Edited customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| DeletedDateUTCUnix | int64 |  | ☐ |  |
| Description | string | Description | ☐ |  |
| Disallow Merge | boolean | Disallow Merge | ☐ |  |
| Edited | datetime | Edited | ☐ |  |
| EditedUTCUnix | int64 |  | ☐ |  |
| Email | string | Email | ☐ |  |
| FirstName | string | First Name | ☐ |  |
| Gender | int32 | Gender | ☐ |  |
| HomePhone | string | Home Phone | ☐ |  |
| HouseExtension | string | House Extension | ☐ |  |
| HouseNumber | string | House Number | ☐ |  |
| Id | int32 | ID | ☐ |  |
| Initials | string | Initials | ☐ |  |
| IsAnonymized | boolean | Is Anonymized | ☐ |  |
| IsDeleted | boolean | Is Deleted | ☐ |  |
| LanguageId | int32 | Language ID | ☐ |  |
| LastName | string | Last Name | ☐ |  |
| LastNamePrefix | string | Last Name Prefix | ☐ |  |
| Latitude | double | Latitude | ☐ |  |
| Longitude | double | Longitude | ☐ |  |
| Mobile | string | Mobile | ☐ |  |
| Name | string | Name | ☐ |  |
| SSN | string | SSN | ☐ |  |
| StreetName | string | Street Name | ☐ |  |
| Title | string | Title | ☐ |  |
| WorkPhone | string | Work Phone | ☐ |  |
| ZipCode | string | ZIP-code | ☐ |  |

### 3.1.15 CustomerSubscriptionSalesByCustomerId: Tixly Customer Subscription Sales by Customer ID

Fetches the subscriptions from a specific customer

Catalog: Tixly

Schema: Customer

Label: Customer Subscription Sales by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerSubscriptionSalesByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| CustomerName | string | Customer Name | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| SubscriptionTypeId | int32 | Subscription Type ID | ☐ | |

### 3.1.16 CustomerTags: Tixly Customer Tags

Fetches all customer tag to customer connections that you have access to. This can be used with the Get all customer tags endpoint and all fetched customer data to quickly connect the two together.

Catalog: Tixly

Schema: Customer

Label: Customer Tags

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| TagId | int32 | Tag ID | ☐ | |

### 3.1.17 CustomerTags2ByCustomerId: Tixly Customer Tags by Customer ID

Fetches all customer tags on a specific customer. This endpoint is not meant to be called once for every customer in the feed, for that purpose it is much more efficient to use the Get all customer tags customer connection endpoint and link them together that way.

Catalog: Tixly

Schema: Customer

Label: Customer Tags by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerTags2ByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | Id of the customer |

## Columns of Table Function

The columns of the table function CustomerTags2ByCustomerId are shown below. Each column has an SQL data type.

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Color | string | Color | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Icon | string | Icon | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

### 3.1.18 CustomerTagsByCustomerId: Tixly Customer Tags by Customer ID

Fetches a specific customer by his customer ID

Catalog: Tixly

Schema: Customer

Label: Customer Tags by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerTagsByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.19 CustomerTicketsByCustomerId: Tixly Customer Tickets by Customer ID

Fetches the tickets a specific customer has bought

Catalog: Tixly

Schema: Customer

Label: Customer Tickets by Customer ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerTicketsByCustomerId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | Id of the customer |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| CustomerId | int32 | Customer ID | ☐ | |
| Entrance | string | Entrance | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Fee | double | Fee | ☐ | |
| IsAnonymousSale | boolean | Is Anonymous Sale | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| PriceZone | string | Price Zone | ☐ | |
| PriceZoneId | int32 | Price Zone ID | ☐ | |
| Row | string | Row | ☐ | |
| Scanned | boolean | Scanned | ☐ | |
| Seat | string | Seat | ☐ | |
| Section | string | Section | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |
| SubscriptionTypeId | int32 | Subscription Type ID | ☐ | |
| TicketCount | int32 | Ticket Count | ☐ | |
| TicketId | int32 | Ticket ID | ☐ | |
| TicketType | string | Ticket Type | ☐ | |
| TicketTypeId | int32 | Ticket Type ID | ☐ | |

### 3.1.20 DeletedCustomerPlusMetadata: Tixly Deleted Customers plus Metadata

Fetches all deleted customers you have access to.

Catalog: Tixly

Schema: Customer

Label: Deleted Customers plus Metadata

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedCustomerPlusMetadata. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Deleted customer from |
| DateTo | datetime | ☐ | | Deleted customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.21 DeletedCustomers: Tixly Deleted Customers

Fetches all deleted customers you have access to.

Catalog: Tixly

Schema: Customer

Label: Deleted Customers

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedCustomers. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Deleted customer from |
| DateTo | datetime | ☐ | | Deleted customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.22 DeletedCustomerTags: Tixly Deleted Customer Tags

Fetches all deleted customers you have access to.

Catalog: Tixly

Schema: Customer

Label: Deleted Customer Tags

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedCustomerTags. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Deleted customer from |
| DateTo | datetime | ☐ | | Deleted customer to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Description | string | Description | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.23 MergedCustomerOperationContents

Gets merged customers details.

Catalog: Tixly

Schema: Customer

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| MasterCustomerId | int32 | Master Customer ID | ☐ | |
| Merged | datetime | Merged | ☐ | |
| MergedUTCUnix | int64 | | ☐ | |
| TEXT | int32 | Text | ☐ | |

### 3.1.24 MergedCustomerOperations

Gets merged customers details.

Catalog: Tixly

Schema: Customer

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| MasterCustomerId | int32 | Master Customer ID | ☐ | |
| Merged | datetime | Merged | ☐ | |
| MergedUTCUnix | int64 | | ☐ | |

### 3.1.25 SaleMetadataRange: Tixly Sale Metadata Range

Fetches customers by sale range, which means if the customer buys or is refunded an order in the date range it will be included. This endpoint is good to use if you are fetching from other endpoints by a specific date range, that can be matched in this endpoint to link the data. The dates are to and from and include time as well so if you would like an entire day then you need to indicate the end time of that day or the start of the next as the DateTo parameter.

Catalog: Tixly

Schema: Customer

Label: Sale Metadata Range

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function SaleMetadataRange. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☑ | | Created sale from |
| DateTo | datetime | ☑ | | Created sale to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.26 SaleRange: Tixly Sale Range

Fetches customers by sale range, which means if the customer buys or is refunded an order in the date range it will be included. This endpoint is good to use if you are fetching from other endpoints by a specific date range, that can be matched in this endpoint to link the data. The dates are to and from and include time as well so if you would like an entire day then you need to indicate the end time of that day or the start of the next as the DateTo parameter.

Catalog: Tixly

Schema: Customer

Label: Sale Range

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

#### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function SaleRange. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☑ | | Created sale from |
| DateTo | datetime | ☑ | | Created sale to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 3.1.27 SaleTagsRange: Tixly Sale Tags Range

Fetches customers by sale range, which means if the customer buys or is refunded an order in the date range it will be included. This endpoint is good to use if you are fetching from other endpoints by a specific date range, that can be matched in this endpoint to link the data. The dates are to and from and include time as well so if you would like an entire day then you need to indicate the end time of that day or the start of the next as the DateTo parameter.

Catalog: Tixly

Schema: Customer

Label: Sale Tags Range

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function SaleTagsRange. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☑ | | Created sale from |
| DateTo | datetime | ☑ | | Created sale to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

# 4 Schema: Donation

## 4.1 Tables

### 4.1.1 DeletedMembershipDonations: Tixly Deleted Membership Donations

Fetches all deleted memberships you have access to.

Catalog: Tixly

Schema: Donation

Label: Deleted Membership Donations

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

#### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedMembershipDonations. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | |
| DateTo | datetime | ☐ | | |

#### Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| Renew DayOfMonth | int32 | Renew Day of Month | ☐ | |
| Renew MonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

### 4.1.2 Donations: Tixly Donations

Fetches all donatons that you have access to, pass in the SoldFrom and SoldTo to get only donations posted on a specific date range.

Catalog: Tixly

Schema: Donation

Label: Donations

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Donations. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Description | string | Description | ☐ | |
| DonationID | int32 | Donation ID | ☐ | |
| Id | int32 | ID | ☐ | |
| isCancelled | boolean | Is Cancelled | ☐ | |
| Name | string | Name | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| RenewalType | string | Renewal Type | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

### 4.1.3 DonationTypes: Tixly Donation Types

Fetches all the donation types you have access to.

Catalog: Tixly

Schema: Donation

Label: Donation Types

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

# 5 Schema: Event

## 5.1 Tables

### 5.1.1 DeletedEvents: Tixly Deleted Events

Fetches all deleted events you have access to.

Catalog: Tixly

Schema: Event

Label: Deleted Events

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedEvents. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Events deleted from |
| DateTo | datetime | ☐ | | Events deleted to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |

### 5.1.2 EventAllocations: Tixly Event Allocations

Fetches all events you have access to.

Catalog: Tixly

Schema: Event

Label: Event Allocations

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventAllocations. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |
| StartFrom | datetime | ☐ | | Get events that start after this date |
| StartTo | datetime | ☐ | | get events that start before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Block | boolean | Block | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| ExpiresHoursBefore | int32 | Expires Hours Before | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| ReservedTickets | int32 | Reserved Tickets | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Seats | int32 | Seats | ☐ | |
| Sold | int32 | Sold | ☐ | |
| SoldTickets | int32 | Sold Tickets | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.3 EventAllocationsByEventId: Tixly Event Allocations by Event ID

Fetches all allocations for a specific events

Catalog: Tixly

Schema: Event

Label: Event Allocations by Event ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventAllocationsByEventId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | date of the specific event |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Block | boolean | Block | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| ExpiresHoursBefore | int32 | Expires Hours Before | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| ReservedTickets | int32 | Reserved Tickets | ☐ | |
| Seats | int32 | Seats | ☐ | |
| SoldTickets | int32 | Sold Tickets | ☐ | |

### 5.1.4 EventCategories: Tixly Event Categories

Fetches all events you have access to.

Catalog: Tixly

Schema: Event

Label: Event Categories

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventCategories. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |
| StartFrom | datetime | ☐ | | Get events that start after this date |
| StartTo | datetime | ☐ | | get events that start before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| OnlineReservationAllow ed | boolean | Online Reservation Allow ed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.5 EventCategories1: Tixly Event Categories

Fetches all events categories you have access to. The categories are returned with a list of IDs of the events they are connected to.

Catalog: Tixly

Schema: Event

Label: Event Categories

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

## Table Columns

The columns of the table EventCategories1 are shown below. Each column has an SQL data type.

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

### 5.1.6 EventCategoriesData: Tixly Event Categories Data

Fetches all events categories you have access to. The categories are returned with a list of IDs of the events they are connected to.

Catalog: Tixly

Schema: Event

Label: Event Categories Data

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| TEXT | int32 | Text | ☐ | |

### 5.1.7 EventCustomerPlusMetadataById: Tixly Event Customer plus Metadata by ID

Fetches the customers that have bought tickets in a specific event

Catalog: Tixly

Schema: Event

Label: Event Customer plus Metadata by ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventCustomerPlusMetadataById. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | date of the specific event |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 5.1.8 EventCustomersByEventId: Tixly Event Customers by Event ID

Fetches the customers that have bought tickets in a specific event

Catalog: Tixly

Schema: Event

Label: Event Customers by Event ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventCustomersByEventId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | date of the specific event |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 5.1.9 EventCustomerTagsById: Tixly Customer Tags by ID

Fetches the customers that have bought tickets in a specific event

Catalog: Tixly

Schema: Event

Label: Customer Tags by ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventCustomerTagsById. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | date of the specific event |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 5.1.10 EventDeletedSeasons: Tixly Event Deleted Seasons

Fetches all events categories you have access to. The categories are returned with a list of IDs of the events they are connected to.

Catalog: Tixly

Schema: Event

Label: Event Deleted Seasons

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventDeletedSeasons. 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 |
|---|---|---|---|---|
| dateFrom | datetime | ☐ | | |
| dateTo | datetime | ☐ | | |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| DateFrom | datetime | Date from | ☐ | |
| DateFromUTCUnix | int64 | | ☐ | |
| DateTo | datetime | Date to | ☐ | |
| DateToUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

### 5.1.11 Events: Tixly Events

Fetches all events you have access to.

Catalog: Tixly

Schema: Event

Label: Events

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Events. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |
| StartFrom | datetime | ☐ | | Get events that start after this date |
| StartTo | datetime | ☐ | | get events that start before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.12 EventTags: Tixly Event Tags

Fetches all events you have access to.

Catalog: Tixly

Schema: Event

Label: Event Tags

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventTags. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |
| StartFrom | datetime | ☐ | | Get events that start after this date |
| StartTo | datetime | ☐ | | get events that start before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.13 EventTags1: Tixly Event Tags

Fetches all events tags you have access to. The tags are returned with a list of IDs of the events they are connected to.

Catalog: Tixly

Schema: Event

Label: Event Tags

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

## Table Columns

The columns of the table EventTags1 are shown below. Each column has an SQL data type.

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

### 5.1.14 EventTagsData: Tixly Event Tags Data

Fetches all events tags you have access to. The tags are returned with a list of IDs of the events they are connected to.

Catalog: Tixly

Schema: Event

Label: Event Tags Data

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| TEXT | int32 | Text | ☐ | |

### 5.1.15 EventTickets: Tixly Event Tickets

Fetches all the tickets that you have access to, pass in the SoldFrom and SoldTo to get tickets sold within a specific date range

Catalog: Tixly

Schema: Event

Label: Event Tickets

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventTickets. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Entrance | string | Entrance | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Fee | double | Fee | ☐ | |
| IsAnonymousSale | boolean | Is Anonymous Sale | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| PriceZone | string | Price Zone | ☐ | |
| PriceZoneId | int32 | Price Zone ID | ☐ | |
| Row | string | Row | ☐ | |
| Scanned | boolean | Scanned | ☐ | |
| Seat | string | Seat | ☐ | |
| Section | string | Section | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |
| SubscriptionTypeId | int32 | Subscription Type ID | ☐ | |
| TicketCount | int32 | Ticket Count | ☐ | |
| TicketId | int32 | Ticket ID | ☐ | |
| TicketType | string | Ticket Type | ☐ | |
| TicketTypeId | int32 | Ticket Type ID | ☐ | |

### 5.1.16 EventTicketsByEventId: Tixly Event Tickets by Event ID

Fetches tickets sold to a specific event

Catalog: Tixly

Schema: Event

Label: Event Tickets by Event ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventTicketsByEventId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | date of the specific event |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Entrance | string | Entrance | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Fee | double | Fee | ☐ | |
| IsAnonymousSale | boolean | Is Anonymous Sale | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| PriceZone | string | Price Zone | ☐ | |
| PriceZoneId | int32 | Price Zone ID | ☐ | |
| Row | string | Row | ☐ | |
| Scanned | boolean | Scanned | ☐ | |
| Seat | string | Seat | ☐ | |
| Section | string | Section | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |
| SubscriptionTypeId | int32 | Subscription Type ID | ☐ | |
| TicketCount | int32 | Ticket Count | ☐ | |
| TicketId | int32 | Ticket ID | ☐ | |
| TicketType | string | Ticket Type | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| TicketTypeId | int32 | Ticket Type ID | ☐ | |

### 5.1.17 FutureEventAllocations: Tixly Future Event Allocations

Fetches events you have access to that have a start date in the future.

Catalog: Tixly

Schema: Event

Label: Future Event Allocations

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function FutureEventAllocations. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Block | boolean | Block | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| ExpiresHoursBefore | int32 | Expires Hours Before | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| ReservedTickets | int32 | Reserved Tickets | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Seats | int32 | Seats | ☐ | |
| Sold | int32 | Sold | ☐ | |
| SoldTickets | int32 | Sold Tickets | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.18 FutureEventCategories: Tixly Future Event Categories

Fetches events you have access to that have a start date in the future.

Catalog: Tixly

Schema: Event

Label: Future Event Categories

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function FutureEventCategories. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllow ed | boolean | Online Reservation Allow ed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.19 FutureEvents: Tixly Future Events

Fetches events you have access to that have a start date in the future.

Catalog: Tixly

Schema: Event

Label: Future Events

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function FutureEvents. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 5.1.20 FutureEventTags: Tixly Future Event Tags

Fetches events you have access to that have a start date in the future.

Catalog: Tixly

Schema: Event

Label: Future Event Tags

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function FutureEventTags. 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 |
|---|---|---|---|---|
| EditedFrom | datetime | ☐ | | Get events edited after this date |
| EditedTo | datetime | ☐ | | Get events edited before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllow ed | boolean | Online Reservation Allow ed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

# 6 Schema: EventGroup

## 6.1 Tables

### 6.1.1 DeletedEventGroups: Tixly Deleted Event Groups

Fetches all deleted event groups you have access to.

Catalog: Tixly

Schema: EventGroup

Label: Deleted Event Groups

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedEventGroups. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Event groups deleted from |
| DateTo | datetime | ☐ | | Event groups deleted to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AvailableOnline | boolean | Available Online | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Name | string | Name | ☐ | |
| Season | string | Season | ☐ | |
| SubTitle | string | Subtitle | ☐ | |

### 6.1.2 EventGroupById: Tixly Event Group by ID

Gets a specific event group by it's ID

Catalog: Tixly

Schema: EventGroup

Label: Event Group by ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventGroupById. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | ID of the specific event group |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Name | string | Name | ☐ | |
| Season | string | Season | ☐ | |
| SubTitle | string | Subtitle | ☐ | |

### 6.1.3 EventGroupEventAllocationsByEventGroupId: Tixly Event Group Allocations by Event Group ID

Gets all the events within the event group

Catalog: Tixly

Schema: EventGroup

Label: Event Group Allocations by Event Group ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventGroupEventAllocationsByEventGroupId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | ID of the specific event group |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Block | boolean | Block | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| ExpiresHoursBefore | int32 | Expires Hours Before | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| ReservedTickets | int32 | Reserved Tickets | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Seats | int32 | Seats | ☐ | |
| Sold | int32 | Sold | ☐ | |
| SoldTickets | int32 | Sold Tickets | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 6.1.4 EventGroupEventCategoriesByEventGroupId: Tixly Event Group Categories by Event Group ID

Gets all the events within the event group

Catalog: Tixly

Schema: EventGroup

Label: Event Group Categories by Event Group ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventGroupEventCategoriesByEventGroupId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | ID of the specific event group |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 6.1.5 EventGroupEventsByEventGroupId: Tixly Event Group Events by Event Group ID

Gets all the events within the event group

Catalog: Tixly

Schema: EventGroup

Label: Event Group Events by Event Group ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventGroupEventsByEventGroupId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | ID of the specific event group |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 6.1.6 EventGroupEventTagsByEventGroupId: Tixly Event Group Tags by Event Group ID

Gets all the events within the event group

Catalog: Tixly

Schema: EventGroup

Label: Event Group Tags by Event Group ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function EventGroupEventTagsByEventGroupId. 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 |
|---|---|---|---|---|
| id | int32 | ✓ | | ID of the specific event group |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Allocated | int32 | Allocated | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| BlockedAllocated | int32 | Blocked Allocated | ☐ | |
| Capacity | int32 | Capacity | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| DateConfirmed | boolean | Date Confirmed | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EndDate | datetime | End Date | ☐ | |
| EndDateUTCUnix | int64 | | ☐ | |
| EventGroup | string | Event Group | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Hall | string | Hall | ☐ | |
| HallId | int32 | Hall ID | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsFreeEvent | boolean | Is Free Event | ☐ | |
| IsNumbered | boolean | Is Numbered | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineReservationAllowed | boolean | Online Reservation Allowed | ☐ | |
| Promoter | string | Promoter | ☐ | |
| PromoterId | int32 | Promoter ID | ☐ | |
| Reserved | int32 | Reserved | ☐ | |
| SaleStatusId | int32 | Sale Status ID | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| Sold | int32 | Sold | ☐ | |
| StartDate | datetime | Start Date | ☐ | |
| StartDateUTCUnix | int64 | | ☐ | |
| SubTitle | string | Subtitle | ☐ | |
| Vat | double | VAT | ☐ | |
| Venue | string | Venue | ☐ | |
| VenueId | int32 | Venue ID | ☐ | |

### 6.1.7 EventGroups: Tixly Event Groups

Gets all the event groups this workgroup has access to.

Catalog: Tixly

Schema: EventGroup

Label: Event Groups

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| EventGroupId | int32 | Event Group ID | ☐ | |
| Name | string | Name | ☐ | |
| Season | string | Season | ☐ | |
| SubTitle | string | Subtitle | ☐ | |

# 7 Schema: Giftcard

## 7.1 Tables

### 7.1.1 GiftCardRange: Tixly Gift Card Range

Fetches all giftcard sales, pass in the SoldFrom and SoldTo query parameters to get giftcards from a specific daterange

Catalog: Tixly

Schema: Giftcard

Label: Gift Card Range

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function GiftCardRange. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Amount | double | Amount | ☐ | |
| AmountLeft | double | Amount Left | ☐ | |
| AmountUsed | double | Amount Used | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Expires | datetime | Expires | ☐ | |
| ExpiresUTCUnix | int64 | | ☐ | |
| GiftCardCount | int32 | Giftcard Count | ☐ | |
| Number | string | Number | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Pin | string | PIN | ☐ | |
| Price | double | Price | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |

# 8 Schema: Membership

## 8.1 Tables

### 8.1.1 DeletedMemberships: Tixly Deleted Memberships

Fetches all deleted memberships you have access to.

Catalog: Tixly

Schema: Membership

Label: Deleted Memberships

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedMemberships. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | |
| DateTo | datetime | ☐ | | |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

### 8.1.2 MembershipDonationTypes: Tixly Membership Donation Types

Fetches all membership types you have access to.

Catalog: Tixly

Schema: Membership

Label: Membership Donation Types

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| Renew DayOfMonth | int32 | Renew Day of Month | ☐ | |
| Renew MonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

### 8.1.3 Memberships: Tixly Memberships

Fetches all memberships that you have access to, pass in the SoldFrom and SoldTo to get only memberships sold on a specific date range.

Catalog: Tixly

Schema: Membership

Label: Memberships

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Memberships. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Description | string | Description | ☐ | |
| Expires | datetime | Expires | ☐ | |
| ExpiresUTCUnix | int64 | | ☐ | |
| Id | int32 | ID | ☐ | |
| isCancelled | boolean | Is Cancelled | ☐ | |
| MembershipID | int32 | Membership ID | ☐ | |
| Name | string | Name | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| RenewalType | string | Renewal Type | ☐ | |
| RenewDayOfMonth | int32 | Renew Day of Month | ☐ | |
| RenewMonthOfYear | int32 | Renew Month of Year | ☐ | |
| ShortDescription | string | Short Description | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |
| Tagline | string | Tagline | ☐ | |
| Vat | double | VAT | ☐ | |

# 9 Schema: Native

## 9.1 Tables

### 9.1.1 NATIVEPLATFORMSCALARREQUESTS: Tixly Native Platform Scalar Requests

{res:itgen_native_platform_scalar_requests_desc}

Catalog: Tixly

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 Tixly 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 Tixly API server.

Retrieve: true

Insert: true

Update: false

Delete: false

## View Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| BLOB_PREFERRED | boolean | BLOB Preferred | ☑ | 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 | string(240) | Content Type | ☐ | |
| DATE_ENDED | datetime | End Date | ☑ | |
| DATE_STARTED | datetime | Start Date | ☑ | |
| DRY_RUN | boolean | Run w ithout Actions | ☑ | |
| DURATION_MS | int32 | Duration (ms) | ☑ | |
| ERROR_MESSAGE_CODE | string(30) | Error Message Code | ☐ | |
| ERROR_MESSAGE_TEXT | string(32000) | Error Message Text | ☐ | |
| FAIL_ON_ERROR | boolean | 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. |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| HTTP_DISK_CACHE_SAVE | boolean | Save HTTP Disk Cache | ☐ | Whether results can be stored in HTTP disk cache. |
| HTTP_DISK_CACHE_USE | boolean | 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 | boolean | Save HTTP Memory Cache | ☐ | Whether results can be stored in HTTP memory cache. |
| HTTP_MEMORY_CACHE_USE | boolean | Use HTTP Memory Cache | ☐ | Whether results can be fetched from HTTP memory cache. |
| HTTP_METHOD | string(30) | HTTP Method | ☐ | |
| HTTP_STATUS_CODE | int16 | HTTP Status Code | ☐ | |
| ORIG_SYSTEM_GROUP | string(4000) | Original System Group | ☐ | |
| ORIG_SYSTEM_REFERENCE | string(4000) | Original System Reference | ☐ | |
| PAYLOAD_TEXT | string | Payload | ☐ | |
| RESULT_BLOB | byte[] | Result BLOB | ☐ | |
| RESULT_DATE_TIME_UTC | datetime | Result Date Time | ☐ | |
| RESULT_NUMBER | decimal | Result Number | ☐ | |
| RESULT_TEXT | string | Result Text | ☐ | |
| SUCCESSFUL | boolean | Succesful | ✓ | |
| TIMEOUT_SEC | int32 | Timeout (sec) | ☐ | Timeout in seconds. |
| TRANSACTION_ID | int32 | Transaction ID | ✓ | Incrementing ID of the transaction. |
| URL | string(4000) | URL | ☐ | |

# 10 Schema: Permission

## 10.1 Tables

### 10.1.1 DeletedPermissions: Tixly Deleted Permissions

Fetches all deleted permission types.

Catalog: Tixly

Schema: Permission

Label: Deleted Permissions

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

#### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedPermissions. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Deleted permisson types from |
| DateTo | datetime | ☐ | | Deleted permission types to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Deleted | datetime | Deleted | ☐ | |
| DeletedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| EmailPermission | boolean | Email Permission | ☐ | |
| Id | int32 | ID | ☐ | |
| MobilePermission | boolean | Mobile Permission | ☐ | |
| Name | string | Name | ☐ | |
| Title | string | Title | ☐ | |
| Type | int32 | Type | ☐ | |

### 10.1.2 PermissionRegistrationsByPermissionId: Tixly Permission Registrations by Permission ID

Fetches all registrations to the specified permission.

Catalog: Tixly

Schema: Permission

Label: Permission Registrations by Permission ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function PermissionRegistrationsByPermissionId. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | If this is set, then only get registrations after this date |
| DateTo | datetime | ☐ | | If this is set, then only get registrations before this date |
| id | int32 | ☑ | | Id of the permission |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| Email | string | Email | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Id | int32 | ID | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| PermissionId | int32 | Permission ID | ☐ | |

### 10.1.3 PermissionRemovalsByPermissionId: Tixly Permission Removals by Permission ID

Fetches all removals from the specified permission.

Catalog: Tixly

Schema: Permission

Label: Permission Removals by Permission ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function PermissionRemovalsByPermissionId. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | If this is set, then only get registration removals after this date |
| DateTo | datetime | ☐ | | If this is set, then only get registration removals before this date |
| id | int32 | ☑ | | Id of the permission |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| CustomerId | int32 | Customer ID | ☐ | |
| Email | string | Email | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Id | int32 | ID | ☐ | |
| Mobile | string | Mobile | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| PermissionId | int32 | Permission ID | ☐ | |
| Removed | datetime | Removed | ☐ | |
| RemovedUTCUnix | int64 | | ☐ | |

### 10.1.4 Permissions: Tixly Permissions

Fetches all permissions you have access to.

Catalog: Tixly

Schema: Permission

Label: Permissions

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| EmailPermission | boolean | Email Permission | ☐ | |
| Id | int32 | ID | ☐ | |
| MobilePermission | boolean | Mobile Permission | ☐ | |
| Name | string | Name | ☐ | |
| Title | string | Title | ☐ | |
| Type | int32 | Type | ☐ | |

# 11 Schema: Product

## 11.1 Tables

### 11.1.1 Products

Fetches all products you have access to.

Catalog: Tixly

Schema: Product

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Products. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| GroupingText | string | Grouping Text | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Price | double | Price | ☐ | |
| Vat | double | VAT | ☐ | |

### 11.1.2 SoldProducts: Tixly Sold Products

Get all the sold products. Pass in the SoldFrom and SoldTo parameters to get sales within a specific date range

Catalog: Tixly

Schema: Product

Label: Sold Products

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function SoldProducts. 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 |
|---|---|---|---|---|
| SoldFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| SoldTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| EventId | int32 | Event ID | ☐ | |
| Online | boolean | Online | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| Price | double | Price | ☐ | |
| ProductCount | int32 | Product Count | ☐ | |
| ProductId | int32 | Product ID | ☐ | |
| SkinID | int32 | Skin ID | ☐ | |

# 12 Schema: Subscription

## 12.1 Tables

### 12.1.1 Subscriptions: Tixly Subscriptions

Gets all subscription sales that you have access to

Catalog: Tixly

Schema: Subscription

Label: Subscriptions

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function Subscriptions. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | If this is set, then only get sales after this date |
| DateTo | datetime | ☐ | | If this is set, then only get sales before this date |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CustomerId | int32 | Customer ID | ☐ | |
| CustomerName | string | Customer Name | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| OrderId | int32 | Order ID | ☐ | |
| OrganisationID | int32 | Organization ID | ☐ | |
| SubscriptionTypeId | int32 | Subscription Type ID | ☐ | |

### 12.1.2 SubscriptionTypeEvents: Tixly Subscription Type Events

Gets all subscription types that you have access to

Catalog: Tixly

Schema: Subscription

Label: Subscription Type Events

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allow Change | boolean | Allow Change | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| Description | string | Description | ☐ | |
| Hall | string | Hall | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| IsConcert | boolean | Is Concert | ☐ | |
| MaxEvents | int32 | Maximum Events | ☐ | |
| MaxTickets | int32 | Maximum Tickets | ☐ | |
| MinEvents | int32 | Minimum Events | ☐ | |
| MinTickets | int32 | Minimum Tickets | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| OnlineSaleEnd | datetime | Online Sale End | ☐ | |
| OnlineSaleStart | datetime | Online Sale Start | ☐ | |
| PickPerEvent | boolean | Pick per Event | ☐ | |
| Renew alEndDate | datetime | Renew al End Date | ☐ | |
| Renew alStartDate | datetime | Renew al Start Date | ☐ | |
| ResourceName | string | Resource Name | ☐ | |
| SameSeat | boolean | Same Seat | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |
| SubTitle | string | Subtitle | ☐ | |

### 12.1.3 SubscriptionTypes: Tixly Subscription Types

Gets all subscription types that you have access to

Catalog: Tixly

Schema: Subscription

Label: Subscription Types

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Allow Change | boolean | Allow Change | ☐ | |
| AvailableOnline | boolean | Available Online | ☐ | |
| Created | datetime | Created | ☐ | |
| Description | string | Description | ☐ | |
| Hall | string | Hall | ☐ | |
| Id | int32 | ID | ☐ | |
| IsConcert | boolean | Is Concert | ☐ | |
| MaxEvents | int32 | Maximum Events | ☐ | |
| MaxTickets | int32 | Maximum Tickets | ☐ | |
| MinEvents | int32 | Minimum Events | ☐ | |
| MinTickets | int32 | Minimum Tickets | ☐ | |
| Name | string | Name | ☐ | |
| OnlineSaleEnd | datetime | Online Sale End | ☐ | |
| OnlineSaleStart | datetime | Online Sale Start | ☐ | |
| PickPerEvent | boolean | Pick per Event | ☐ | |
| Renew alEndDate | datetime | Renew al End Date | ☐ | |
| Renew alStartDate | datetime | Renew al Start Date | ☐ | |
| ResourceName | string | Resource Name | ☐ | |
| SameSeat | boolean | Same Seat | ☐ | |
| Season | string | Season | ☐ | |
| SeasonId | int32 | Season ID | ☐ | |

# 13 Schema: Tag

## 13.1 Tables

### 13.1.1 CustomerDataByTagId: Tixly Customer Data by Tag ID

Gets all customers that have a specific customer tag

Catalog: Tixly

Schema: Tag

Label: Customer Data by Tag ID

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

#### Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerDataByTagId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the specific customer tag to get |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| Color | string | Color | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created_1 | datetime | Created 1 | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix_1 | int64 | | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Icon | string | Icon | ☐ | |
| Id_1 | int32 | ID 1 | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 13.1.2 CustomerMetadataByTagId: Tixly Customer Metadata by Tag ID

Gets all customers that have a specific customer tag

Catalog: Tixly

Schema: Tag

Label: Customer Metadata by Tag ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomerMetadataByTagId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the specific customer tag to get |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| CustomerAttributeID | int32 | Customer Atribute ID | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| DisallowMerge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name_1 | string | Name 1 | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| Value | string | Value | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 13.1.3 CustomersByTagId: Tixly Customers by Tag ID

Gets all customers that have a specific customer tag

Catalog: Tixly

Schema: Tag

Label: Customers by Tag ID

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function CustomersByTagId. 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 |
|---|---|---|---|---|
| id | int32 | ☑ | | Id of the specific customer tag to get |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| AddressOne | string | Address One | ☐ | |
| AddressTwo | string | Address Two | ☐ | |
| AlternativeEmails | string | Alternative Email Addresses | ☐ | |
| City | string | City | ☐ | |
| CompanyName | string | Company Name | ☐ | |
| Country | string | Country | ☐ | |
| CountryCode | string | Country Code | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| CRN | string | CRN | ☐ | |
| DateOfBirth | datetime | Date of Birth | ☐ | |
| DateOfBirthUTCUnix | int64 | | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Description | string | Description | ☐ | |
| Disallow Merge | boolean | Disallow Merge | ☐ | |
| Edited | datetime | Edited | ☐ | |
| EditedUTCUnix | int64 | | ☐ | |
| Email | string | Email | ☐ | |
| FirstName | string | First Name | ☐ | |
| Gender | int32 | Gender | ☐ | |
| HomePhone | string | Home Phone | ☐ | |
| HouseExtension | string | House Extension | ☐ | |
| HouseNumber | string | House Number | ☐ | |
| Id | int32 | ID | ☐ | |
| Initials | string | Initials | ☐ | |
| IsAnonymized | boolean | Is Anonymized | ☐ | |
| IsDeleted | boolean | Is Deleted | ☐ | |
| LanguageId | int32 | Language ID | ☐ | |
| LastName | string | Last Name | ☐ | |
| LastNamePrefix | string | Last Name Prefix | ☐ | |
| Latitude | double | Latitude | ☐ | |

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Longitude | double | Longitude | ☐ | |
| Mobile | string | Mobile | ☐ | |
| Name | string | Name | ☐ | |
| SSN | string | SSN | ☐ | |
| StreetName | string | Street Name | ☐ | |
| Title | string | Title | ☐ | |
| WorkPhone | string | Work Phone | ☐ | |
| ZipCode | string | ZIP-code | ☐ | |

### 13.1.4 DeletedTags: Tixly Deleted Tags

Fetches all deleted customer tags you have access to.

Catalog: Tixly

Schema: Tag

Label: Deleted Tags

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedTags. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | Deleted customer tag from |
| DateTo | datetime | ☐ | | Deleted customer tag to |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Color | string | Color | ☐ | |
| Deleted | datetime | Deleted | ☐ | |
| DeletedUTCUnix | int64 | | ☐ | |
| Icon | string | Icon | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

#### 13.1.5 Tags: Tixly Tags

Fetches all customer tags that you have access to.

Catalog: Tixly

Schema: Tag

Label: Tags

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Abbreviation | string | Abbreviation | ☐ | |
| Color | string | Color | ☐ | |
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Icon | string | Icon | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |

# 14 Schema: Ticket

## 14.1 Tables

#### 14.1.1 DeletedTicketTypes: Tixly Deleted Ticket Types

Fetches all deleted ticket types you have access to.

Catalog: Tixly

Schema: Ticket

Label: Deleted Ticket Types

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

## Parameters of Table Function

The following parameters can be used to control the behaviour of the table function DeletedTicketTypes. 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 |
|---|---|---|---|---|
| DateFrom | datetime | ☐ | | |
| DateTo | datetime | ☐ | | |

## Columns of Table Function

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Default | boolean | Default | ☐ | |
| DeletedDate | datetime | Deleted Date | ☐ | |
| DeletedDateUTCUnix | int64 | | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Position | int32 | Position | ☐ | |
| TicketTypeGroupId | int32 | Ticket Type Group ID | ☐ | |
| Type | int32 | Type | ☐ | |
| TypeName | string | Type Name | ☐ | |

### 14.1.2 TicketTypeGroups: Tixly Ticket Type Groups

Fetches all ticket type groups you have access to.

Catalog: Tixly

Schema: Ticket

Label: Ticket Type Groups

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Position | int32 | Position | ☐ | |

### 14.1.3 TicketTypes: Tixly Ticket Types

Fetches all ticket types you have access to.

Catalog: Tixly

Schema: Ticket

Label: Ticket Types

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

## Table Columns

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

| Name | Data Type | Label | Required | Documentation |
|---|---|---|---|---|
| Created | datetime | Created | ☐ | |
| CreatedUTCUnix | int64 | | ☐ | |
| Default | boolean | Default | ☐ | |
| Id | int32 | ID | ☐ | |
| Name | string | Name | ☐ | |
| Position | int32 | Position | ☐ | |
| TicketTypeGroupId | int32 | Ticket Type Group ID | ☐ | |
| Type | int32 | Type | ☐ | |
| TypeName | string | Type Name | ☐ | |

# Index

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

23.0.92
15/01/2025, 20:33
ISBN 978-94-6302-000-8: