API Usage Limits

API Name	Limits and Best Practices for Usage
Stream API	To provide consistent performance and reliability at scale, Unity strongly recommends that the customers use the following when planning data ingestion and while using the Stream API • *Data Entities APIs are suited for continuously ingesting data into the platform for specific updates for the staging tables and are not designed for data ingestion at a very high throughput. Customers can use these API's and no limits would be imposed on the number of records sent in using these API's as long as they are within the 'reasonable use" which is defined as a max throughput of • Entities API: 1500 Records/Sec/Customer (Soft Limit) - Feature Releasing soon • Entities API Hard Limit: 2000 Records/Sec/Customer (System (Hard) Limit) - Feature Available today • The recommend message size is 10 records per API call. Sending two calls with 20 records concurrently is generally faster than sending one call with 40 records. We recommend sending small concurrent requests is generally faster than sending one call with a large number of records. • Customers may within a 60 sec period for all sources (in aggregate) should not send more than 90,000 records (Soft Limit), if Unity does receive more data than indicated within that time period it might lead to unplanned delays in receipt, processing and availability of such data. • API calls which exceed these limits might lead to unplanned delays in receipt, processing, and availability of such data. Unity also reserves the right to reject API calls with 429 error upon which additional data sent would be rejected or queue any additional events and process those at a slower rate which ensures quality of service for all customers. • Any client who has a use case to send us more than the above number needs to get in touch with Oracle support for subsequent approval and enabling processing of such payloads. • Customers who have very large streaming needs will need extra preparation and a longer approval process, such requests **might not be accommodated based on operational considerations.** • Unity uses a distributed architecture and is scalable for reasonable data loads. Each customer has different use cases, data ingestion needs and Unity will make best case efforts to receive incoming data loads assuming typical operational situations. For specialized data shapes or very large volumes of more than 2.7 million records to be sent over a period of an 30 mins please contact Oracle support. • Once Unity receives the data, we will process the data payloads in our platform with best effort, Event bursts that exceed the payload size or throughput volume may result in queued processing. *Under typical operational situations** latency will vary based on the ingestion method used, kind of data objects being updated, use cases, data model and other operational factors. ** Large volume data ingestion ingestion for Unity should primarily be done using Ingest jobs, these jobs provide scale and speed required for high data volume ingestion. Use cases where customers need to upload significant amount of data regularly should only be done using these Ingest Jobs. We will process these jobs using high volume parallel processing techniques*
Near Real time API	To provide consistent performance and reliability at scale, Unity strongly recommends that the customers use the following when planning data ingestion and while using Near Real time API. • *Real-Time APIs are suited for continuously ingesting real-time data streams into the platform for specific updates and not data ingestion at scale. Customers can use these API's and no limits would be imposed on the number of records sent in using these API's as long as they are within the 'reasonable use" which is defined as a max throughput of • Streams API: 2000 Records/Sec/Customer (Soft Limit) - Feature Releasing soon • Streams API: 4000 Records/Sec/Customer (Hard Limit) - Feature Available today • The recommend message size is 10 records per API call. Sending two calls with 20 records concurrently is generally faster than sending one call with 40 records. Consequently, we recommend sending small concurrent requests is generally faster than sending one call with a large number of records. • Customers may within a 60 sec period for all sources (in aggregate) should not send more than 120,000 records (Soft Limit), if Unity does receive more data within that time period it might lead to unplanned delays in receipt, processing and availability of such data. • API calls which exceed these limits might lead to unplanned delays in receipt, processing, and availability of such data. Unity also reserves the right to reject API calls with 429 error upon which additional data sent would be rejected or queue any additional events and process those at a slower rate which ensures quality of service for all customers. • Any client who has a use case to send us more than the above number needs to get in touch with Oracle support for subsequent approval and enabling processing of such payloads. • Customers who have very large streaming needs will need extra preparation and a longer approval process, such requests **might not be accommodated based on operational considerations.** • Unity uses a distributed architecture and is scalable for reasonable data loads. Each customer has different use cases, data ingestion needs and Unity will make best case efforts to receive incoming data loads assuming typical operational situations. For specialized data shapes or very large volumes of more than 3.6 million records to be sent over a period of an 30 mins please contact Oracle support. • Once Unity receives the data, we will process the data payloads in our platform with best effort, Event bursts that exceed the payload size or throughput volume may result in queued processing. *Under typical operational situations** latency will vary based on the ingestion method used, kind of data objects being updated, use cases, data model and other operational factors. ** Large volume data ingestion ingestion for Unity should primarily be done using Ingest jobs, these jobs provide scale and speed required for high data volume ingestion. Use cases where customers need to upload significant amount of data regularly should only be done using these Ingest Jobs. We will process these jobs using high volume parallel processing techniques*
API Concurrency	Clients ideally should keep the API Calls at a maximum of 10 concurrent calls/Tenant.

Data Processing Latency:	• Once Unity receives the data, we will process the data payloads in our platform with best effort, Event bursts that exceed the payload size or throughput volume may result in queued processing. Under typical operational situations* latency will vary based on the ingestion method used, kind of data objects being updated, use cases, data model and other factors. • near real time streamed updates are typically processed within 5-10 under normal circumstances*. • idgraph profile updates are typically processes within 5-10 minutes under normal circumstances*.