[This is unreviewed, DRAFT content on the Performance Analyzer Export file format. This will serve as the basis for public documentation.]

**Overview**

The Performance Analyzer captures operations that occur while a user interacts with a report. The Performance Analyzer pane summarizes those operations so users can see basic information about each visual that is updated. The Performance Analyzer pane also allows users to export the captured performance data. To support more in-depth analysis, the export file contains the underlying performance events, rather than just the summary information shown in the pane.

**Background on Power BI Report Execution**

The Performance Analyzer captures operations that occur in several major subsystems involved in executing a Power BI Report:

* **Report Canvas** provides the user interface for Power BI reports including hosting visuals and filters, managing user interactions for consuming and authoring reports, and retrieving data for display. The Report Canvas is written using web technologies and runs in web browsers or web browser components. The Report Canvas retrieves data using a high-level, internal, Power BI query language known as Semantic Query.
* **Data Shape Engine (DSE)** evaluates Semantic Queries by generating and running one, or more DAX queries against a data model hosted inside Power BI, Power BI Desktop, Azure Analysis Services, or SQL Server Analysis Services.
* **Data Model Engine (AS)** stores the data model and provides services to reports, such as DAX queries evaluation. The model may be hosted in Power BI, Power BI Desktop, Azure Analysis Services, or SQL Server Analysis Services. Depending on the data model host, a model may be tabular or multidimensional. Tabular models may contain in-memory tables, Direct Query tables, or a mix of such tables. DAX queries against tables in Direct Query mode will trigger queries to the Direct Query data source. For example, a DAX query against a Direct Query table backed by a SQL Server database will trigger one, or more, SQL queries.

**Relating User Actions to Visuals and Queries**

When using the Performance Analyzer, or looking at a performance export, it is helpful to understand the relationship between user actions visuals and queries. The following diagram shows how each concept is related.

User Action 
1 Upda 
Visual 
Runs 
Sernantic 
Que 
Runs 
DAX 
Runs 
Direct Query 
Source Query 

A **user action** is any interaction with a report, such as switching pages or changing a filter. Each user action may require one, or more, visuals to be updated. For example, changing a page filter will update all visuals on that page. Performing several actions very quickly may only update affected visuals once. For example, rapidly selecting values A and B in a page filter may only update the visuals on the page once.

A **visual** is any object on the reporting canvas such as a chart, table, slicer, or textbox. For each visual being updated, the Report Canvas may run a single Semantic Query to retrieve data. Not all visuals run queries. For example, a shape visual can be rendered without any data. The Report Canvas has an in-memory cache of results for Semantic Queries run in the current session. When a query matches a cached result, the visual will be updated from the cached result rather than running a new query. Some operations in Power BI Desktop, such as model changes, clear this cache. The cache can also be bypassed visual and page level refresh buttons provided by the Performance Analyzer.

**Semantic Query** is a high-level, internal, query language Power BI uses to retrieve structured data for a visual. The Data Shape Engine evaluates Semantic Queries by generating and running one, or more, **DAX queries** against the Data Model Engine (AS). In most cases, a single Semantic Query can be evaluated using a single DAX query. Multiple DAX queries are usually needed for tabular data models hosted in SQL Server Analysis Services 2012 or 2014 or multidimensional cubes.

DAX queries involving tables in Direct Query mode will trigger one, or more, queries against the Direct Query data source.

**Understanding Performance Analyzer Durations**

Durations in the Performance Analyzer are computed as the difference between the start timestamp and the end timestamp of an operation, sometimes called "wall clock" durations. The reported durations include time spent queued while other operations complete. Each Power BI subsystem has its own patterns for executing operations based on implementation details, resource usage, and hardware.

* The Reporting Canvas and visuals executes most operations sequentially on a single UI thread. Updating a visual requires several different Reporting Canvas operations including generating a query, submitting it to the Data Shape Engine for evaluation, reading the query result, and displaying the visual with the new data. When multiple operations need to be performed at the same time, some operations are queued.
* Power BI backend subsystems, including the Data Shape Engine and Data Model Engine (AS) generally execute operations in parallel using multiple threads.

For example, consider a report page containing three visuals. If each visual is updated sequentially by a different user action, no time is spent queueing. (e.g. User action 1 updates Visual 1. Once that is complete user action 2 updates Visual 2. Once that is complete, user action 3 updates Visual 3. The following diagram provided a simplified illustration of the operations performed by the Report Canvas and Backend. Operations are not shown to scale.

1 Tot—I 
3 Tot—I 
l:w.it 
for 
ual tfo 
3:w.it 
Pre* e 
Rep. t C. n 

Each visual is updated separately, and no operation is queued waiting for other operations to complete.

Now, consider a single user action that updates all three visuals. The Report Canvas operations to update each visual will be interleaved because only one operation executes at a time. Queries in the "backend" DSE and AS components generally run in parallel.

The following diagram shows one possible interleaving pattern. The exact order of operations depends on many factors such as the content of the visuals, the kinds of operations, and the user's hardware. Based on this interleaving, the diagram shows when each visual is queued. These queued times are shown in the Performance Analyzer under "Other".

Visual 3 Queued 
Visual 3TotaI 
Visual 3 Queued 
Visual 2 Total 
Visual 2 Queued 
Visual 2 Queued 
Visuals: Visual 
Display 
Frontend 
operations run 
sequentially 
Backend 
operations 
run in parallel 
'v"sual 2 Queued 
Visual I: 
Prepare Query 
Report Canvas 
Backend 
Visua12: 
Pre pare 
Cue 
'"sual ITotaI 
Visual I Queued 
Visual 3: 
Prepare Quer•,' 
Visual 2:Query 
Visual 
Read 
Res ult 
Visual I: Visual 
Display 
Visi_el 
2: Read 
Result 
Visual 
3: Read 
Result 
Visual 2: Visual 
Display 

These diagrams show a very simplified view of how visuals update. In practice, each operation may get sliced into many smaller tasks that are interleaved with other tasks.

**Export File Format**

The export file is a JSON document containing a list of events representing the operations that occurred while the Performance Analyzer was running. Each event contains timestamps, correlation information, and other metadata about the operation.

Logically, the events represent a forest of trees with each tree representing a user action or visual update. Parent and child events can be correlated using the **id** and **parentId** properties on each event.

Eventl 
Event2 
Event3 
Event4 
Event5 
Event6 
Eventl 
Event2 
Event3 
Event4 
Event5 
Event6 

**Schema**

A JSON schema is available in <<TODO link>>.

**Event Structure**

Each event has the following structure.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Required** | **Description** |
| name | string | Yes | Together with 'component', 'name' identifies the kind of event that occurred. |
| component | string | Yes | The subsystem that performed the operation represented by this event. |
| id | string | Yes | The unique identifier for this event. |
| parentId | string | No | The id of this event's parent event. This must match the id of some other event in this document. Omitted for root events. |
| start | string (date-time) | Yes | The start time of the event as a UTC date time value in RFC3339 format. |
| end | string (date-time) | No | The end time of the event as a UTC date time value in RFC3339 format. Omitted for instantaneous events that do not have a duration. |
| metrics |  | No | A property bag of additional information about this event. The content varies for each kind of event.    The possible metrics for each kind of event are described below. |

**Example**

The following shows an example export containing four events:

|  |
| --- |
| {  "version": "1.0.0",  "events": [{  "name": "User Action",  "component": "Report Canvas",  "start": "2019-04-26T19:42:01.991Z",  "id": "ea168828b4c0175aa697",  "metrics": {  "sourceLabel": "UserAction\_Refresh"  }  }, {  "name": "Visual Container Lifecycle",  "component": "Report Canvas",  "start": "2019-04-26T19:42:02.000Z",  "end": "2019-04-26T19:42:02.343Z",  "id": "c2d2e0b2cf6a46628b68",  "metrics": {  "status": "finished",  "visualTitle": "Count of ProductKey by EnglishCountryRegionName"  }  }, {  "name": "Query",  "component": "Report Canvas",  "start": "2019-04-26T19:42:02.001Z",  "end": "2019-04-26T19:42:02.325Z",  "id": "a8d101ca1ff7de189598",  "parentId": "c2d2e0b2cf6a46628b68"  }, {  "name": "Render",  "component": "Report Canvas",  "start": "2019-04-26T19:42:02.325Z",  "end": "2019-04-26T19:42:02.343Z",  "id": "ccc0f88232a87649e78d",  "parentId": "c2d2e0b2cf6a46628b68"  }  ]  } |

**Event Sequencing and Overlap**

The following diagram shows the relationship between performance events included in export. Time flows from left to right, and events are sequenced from left to right. Events that overlap with other events are stacked vertically in the diagram. Events expected to occur multiple times are shown as overlapping boxes.

Report Canvas: 
User Acti on 
Report Canvas: Visual Container Lifecycle 
Report Canvas: Query 
Report Canvas: Render 
Report Canvas:Data 
View Transform 
Process / Network 
Boundary 
BSE: Execute Semantic Query 
DSE: Execute DAX Query 
Process / Network 
Boundary 
AS: Execute Query 
Report Canvas: 
Eeocodin 
AS: Exe cute 
Direct auen/ 
AS : Get Source 
Co nnection 
Process / Network 
Bounda ry 
Di rect Query 
Data Source 
Time 
AS: Serialize 
Rows et 

**Event Cardinality**

The general relationship between user actions, visuals, and queries is described above. However, the actual number and type of events logged for a visual update is dependent on the state of the system, user action, user permissions, data model host, and report configuration. Here are some notable examples:

* No Data Shape Engine (DSE) or Data Model Engine (AS) events are logged if a visual is rendered from the Report Canvas in-memory Semantic Query result cache.
* Data Model Engine (AS) will not appear for models hosted in SQL Server Analysis Services, Power BI, or Azure Analysis Services.
* Direct Query events, such as Execute Direct Query, will only appear for Direct Query or Composite models.

**Timestamp Alignment**

As the diagram above shows, rendering a Power BI report involves multiple processes that may be running on different machines. Event start and end timestamps may not be perfectly aligned across process or machine boundaries due to differences in the system clock. In most cases these differences are small, but they can be significant when crossing machine boundaries.

**Event Details**

The following table describes each kind of event and that event's possible metrics. All metrics are optional.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event Name** | **Component** | **Description** | **Has End** | **Metrics** |
| User Action | Report Canvas | Marks a point where the user interacted with the report. | No | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | sourceLabel | String | The name of the action. | |
| Visual Container Lifecycle | Report Canvas | Tracks an update to a visual. | Yes | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | status | String (enum) | The name of the action.   |  |  | | --- | --- | | **Value** | **Description** | | started | The update is in progress. | | finished | The update completed. | | abandoned | The update was not completed. This can happen if another update started before this update completed. | | | visualTitle | String | The title of the visual. If the visual has no title, this holds the visual type. | |
| Query | Report Canvas | Tracks generating and executing the Semantic Query for a visual. | Yes |  |
| Render | Report Canvas | Tracks preparing the data for the visual and updating the visual with the query result. | Yes |  |
| Data View Transform | Report Canvas | Tracks preparing the data for use by a visual. This includes re-shaping the data, evaluating forecasting and evaluating conditional formatting rules. | Yes |  |
| Geocoding | Report Canvas | Tracks geocoding points for display on a map visual.    Only present if a visual performs geocoding. | Yes |  |
| Execute Semantic Query | DSE | Tracks evaluating a single Semantic Query. | Yes |  |
| Execute DAX Query | DSE | Tracks running a single DAX query against a data model hosted in Power BI Desktop, Power BI, Azure Analysis Services, or SQL Server Analysis Services. The event ends when the first row of the result is received. | Yes | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | QueryText | String | The text of the DAX query. | | RowCount | Number | The number of rows read from the DAX query. | | Error | Boolean | True when the DAX query failed. | | Canceled | Boolean | True when the DAX query was canceled before it could complete. | |
| Metrics Truncated | DSE | Indicates that events were omitted from the performance trace because too many events were generated. Power BI allows up to 300 events per Semantic Query execution and 200 events per DAX query execution. | No |  |
| Execute Query | AS | Tracks evaluating a single query in the data model. | Yes |  |
| Serialize Rowset | AS | Tracks writing the query result. | Yes |  |
| Get Source Connection | AS | Tracks acquiring a connection to the direct query data source.    Only present if the query accessed a Direct Query table. | Yes |  |
| Execute Direct Query | AS | Tracks executing a query against the direct query data source.    Only present if the query accessed a Direct Query table. | Yes | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | QueryText | String | The text of the query.    QueryText is only present if the user owns the data model and the data source is SQL. | | ActualQueryDuration | Number | The time, in milliseconds, spent waiting for the query to execute. | | RowsRead | Number | The number of rows read from this query. | | DataReadDuration | Number | The time, in milliseconds, spent reading the result of this query. | | IsGetSourceCapabilitiesQuery | Boolean | True if this is a metadata query used to fetch the capabilities of the Direct Query data source. | |
| Metrics Truncated | AS | Indicates that events were omitted from the performance trace because too many events were generated. Power BI allows up to 200 events per DAX query execution. | No | |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | Count | Number | The number of events that were omitted. | |