Firebolt Implementation Traits

Firebolt as an API expects the implementation to fulfill certain basic traits as an Application platform.

In this document, we will deep dive into the Firebolt Implementation requirements.

Understanding Firebolt

This section will dive into more details of what traits Firebolt API expects from its implementations

Firebolt APIs

There are mainly 2 kinds of Firebolt APIs

Mandatory API: Below is the list of mandatory APIs expected by the Implementation.

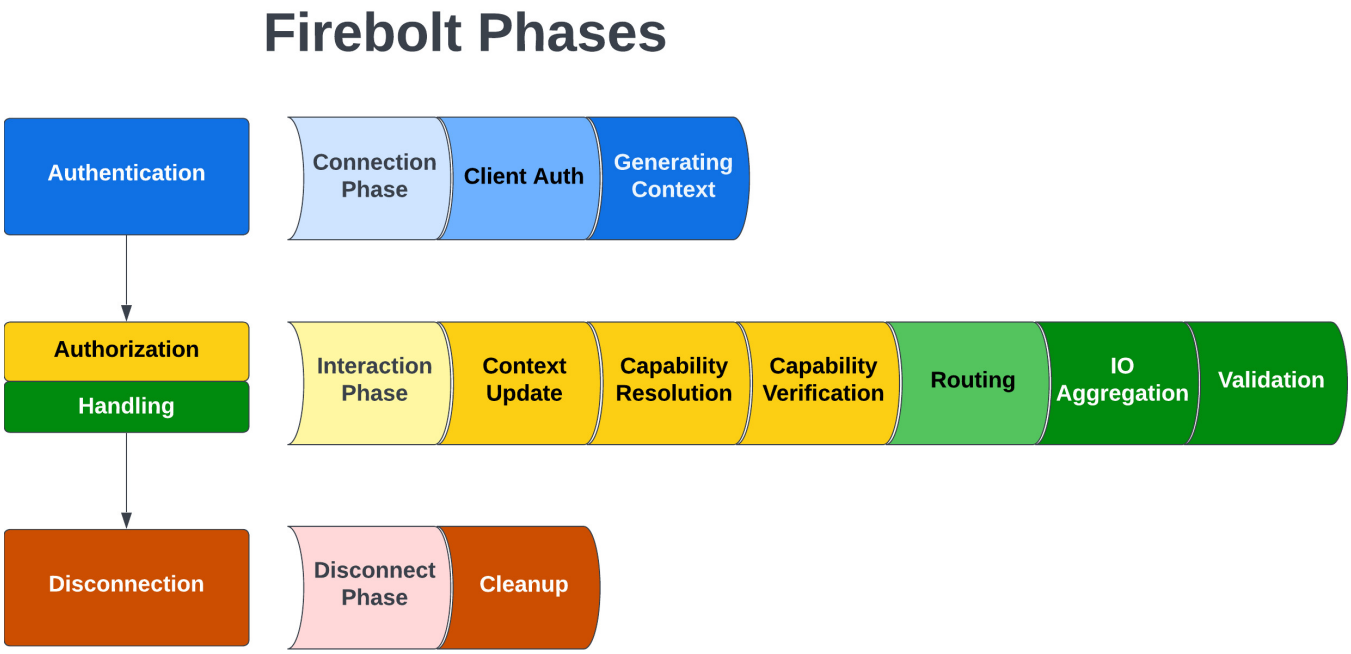
- Capabilities.*
- UserGrants.*
- Lifecycle.*
- LifecycleManagement.*
- Parameters.*

Common API: Any other API which is not part of the above list can be termed as Common APIs

Firebolt Phases

Firebolt shares 3 main phases during its API lifecycle with the Apps.

1. Connection Phase: The App tries to establish a WebSocket connection during this phase and undergoes an Authentication check. The App is fully connected after the successful completion of this step and goes into the next phase.
2. Interaction Phase: The App will now be able to send Firebolt requests and await responses and events. The interaction phase is further divided into stages
    a. Authorization stage: The request is first enriched with contextual metadata. Using the Firebolt Open RPC specification list of capabilities is resolved for a given request. These capabilities are then challenged for Support, Availability, Permission, and User Grants for Authorization. If all the checks are successful the request is fully authorized to proceed to the next stage.
    b. Routing Stage: The request would then need to be resolved to the right handler which will implement the actual Firebolt API.
    c. Handling Stage: Firebolt implementation can either implement the API or broker the implementation like Application to Application provided capabilities.
3. Disconnection Phase: Once the App terminates the WebSocket cleanup needs to be performed.



Let's look into some of these in a bit more detail

Connection Phase:

Firebolt API expects the implementation to be aware of the App.

There could be 2 kinds of Apps connected to the Firebolt Implementation

1. System Apps: Apps that are internal to the platform and have more capabilities compared to 3rd party apps. They connect in a different secure port and have more API(S) available to them in this port.
2. 3rd party Apps: Apps that are hosted on the platform provided by 3rd party, these apps connect on a secure port using the session information and go through Session checks as part of their Authentication protocol.

Context Generation: Implementation is expected to be App aware based on the initial connection as the requests do not have an App ID field. Implementation is expected to create a context for a given connection using the connection parameter. For a System App, this could be a URL parameter, and for 3rd party apps, this could be a session ID generated by the Implementation.

Authorization Stage:

Context Update: Firebolt implementation is expected to be aware of the calling Application from the Connection phase. This data is mandatory for verifying if the given App is authorized for the capabilities to make the request.

Capability Resolution: Firebolt Open RPC specification defines a list of capabilities and their mapping to the request methods in Firebolt. Every implementation is expected to consume a single or multiple Open RPC files to resolve a set of capabilities expected for a given request.

Capability Verification: Each Firebolt request undergoes a rigorous set of checks from supported, available, permitted and user grants to get authorized. This stage expects the implementation to have a list of data sources like manifests, services, and app providers to power the validation.

Routing Stage:

Firebolt Implementation needs a configurable router that forwards the authorized requests to the right handlers. These handlers could be internal to the implementation or external components like Apps, Device Platforms (Thunder), or Cloud services.

Handling Stage:

Firebolt Implementation is expected to resolve the right handler for the request, this routing could be

1. Self-sourced: Firebolt implementation can create its handlers and provide support for the requested API request. This is common for most of the Basic API(s)
2. External-sourced: Firebolt Implementation must have the ability to broker some capabilities to another App or component layer like Thunder. Implementation is still expected to ensure error handling and schema verification are performed during these transactions.

For an Externally sourced API, the implementation is still expected to

1. Validate the schema of the incoming request based on Open RPC spec.
2. Aggregate the responses and events.
3. Validated the outgoing response, and events.

Firebolt Implementation Traits

Based on the above information below is the list of traits expected from a Firebolt Implementation

Firebolt Implementation Authentication Traits

1. Implementation must provide a secure port for 3'rd party apps it can have an internal port for System Apps.
2. Implementation must be connection-aware and match a request to the connecting App.

Firebolt Implementation Authorization Traits

1. Implementation must have the ability to consume multiple Open RPC specification files and create a method for capabilities mapping needed for Capability Resolution.
2. Implementation must have the ability to consume the Device manifest file based on the specified schema
3. Implementation must support configuration to provide a list of supported capabilities provided by the platform.
4. Implementation must implement validation of supported capabilities during the Authorization stage.
5. Implementation must maintain the availability status of a given capability and provide validation during the Authorization stage.
6. Implementation must be able to aggregate the list of capabilities permitted for a given application and provide validation during the Authorization stage.
7. Implementation must be able to understand grant policies, non-negotiable capabilities, cloud mapping for privacy, and other grant strategies to apply user grants to a given request during the Authorization stage.

Firebolt Implementation Routing Traits

1. Implementation must resolve the right handler for a given request and provide backward compatibility for multiple Open RPC specifications.
2. Implementation must be able to forward the request to an external component outside the implementation.

Firebolt Implementation Handling Traits

1. Implementation must have Mandatory API methods created internally and made available for the routing.
2. Implementation must support brokering to applications for provided capabilities.
3. Implementation must allow common APIs to be a pass-through for external components.