**FINOS**

Fintech
Open Source
Foundation

# FDC3

## FDC3 API changes

Proposed by the Channels, Feeds & Transactions discussion group

# Channels, Feeds & Transactions discussion group

- Convened to work on issues related to:
  - API support for Channels and Intents
  - transactions (a standardized way to make a request and receive a response back)
  - feeds (a standardized way to request a stream of data, such as a pricing stream).
- Discussion group
  - No decision making power - proposals brought back to Standards Working Group
  - 9 Meetings held 1st July - 30th November
- 13 PRs raised
  - 5 PRs raised to clarify existing standard documentation (4 merged, one open)
  - 8 PRs raised to enhance the standard (awaiting review)
- **Goal: complete reviews by next meeting (currently 27th Jan 2022) then merge**
  - **Please comment on individual issues/PRs with any feedback**

# Pull requests open for review

## 1. Rename System channels User channels to reflect usage

(issue #430)(PR #479)

**Summary:**

The term **'System Channels'** often generates confusion for new users of the standard, replace it with **'User channels'**

# Pull requests open for review

## 1. Rename System channels User channels to reflect usage

**Changes:**

- Deprecates **`fdc3.getSystemChannels()`** & **`fdc3.joinChannel()`**
- Replaces them with **`fdc3.getUserChannels()`** & **`fdc3.joinUserChannel()`**
- Clarifies channel types through spec and documentation, e.g.:
  - you *join* a User channel and broadcast with **`fdc3.broadcast()`**
  - you **`fdc3.getOrCreateChannel()`** an App channel and broadcast to it directly with **`channel.broadcast()`**.

# Pull requests open for review

## 2. Make it possible to return data from a raised intent

(a transaction)
(issue #432)(PR #495)

**Summary:**

Returning data was part of the original idea for intents (see the FDC3 1.0 spec)
...but the process for doing so was not correctly defined.

Adding support for returning data should not break or change existing 'fire and forget' intents.

Data might be returned after a delay due to processing or user input.

- E.g. a 'raiseOrder' intent might require a user to confirm order details and submit the order, before it returns a context defining the order and its new Id.
- The intent raiser should know which app its waiting on and if any errors occur.

# Pull requests open for review

## 2. Make it possible to return data from a raised intent (cont.)

**Changes:**

- Intent listeners should be able to return data (if they want to):
  - New type **`IntentHandler`** for Intent listeners :
    ```
    type IntentHandler = (context: Context) => Promise<Context> | void;
    ```
- Intent raisers can await a result (if they want to):
  - ```
    export interface IntentResolution {
      readonly source: TargetApp;
      readonly version?: string;
      getResult(): Promise<Context>;
    }
    ```
    ```

# Pull requests open for review

## 2. Make it possible to return data from a raised intent (cont.)

2-step process for resolving intent & receiving data back (separates concerns):

# Pull requests open for review

## 2. Make it possible to return data from a raised intent (cont.)

2-step process for resolving intent & receiving data back (separates concerns):

```
//raise an intent
let resolution = await agent.raiseIntent("intentName", context);

//collect a result
try {
    const result = await resolution.getResult();
    console.log(`${resolution.source} returned ${result}`);
} catch(error) {
    console.error(`${resolution.source} returned an error: ${error}`);
}
```

# Pull requests open for review

## 3. Allow intents to be resolved on output type (where they return data) (issue #498)(PR #499 *)

**Summary:**

If intents can return `Context` data, it should be possible to:

- Resolve them with a requested response type.
- Receive data on what types each application returns in a resolution.

* Incorporates previous PR #495

# Pull requests open for review

## 3. Allow intents to be resolved on output type (cont.)

**Changes:**

Adds an element to the AppD Application schema to specify **`Context`** returned by an intent handler:

```
{
     "name": "Example App",
     "appId": "Example App v1.0",
     …
     "intents": [
          {
               "name": "View Order",
               "displayName": "View Order",
               "contexts": ["fdc3.order"],
               "resultContext": "fdc3.order"
          }
     ]
}
```

# Pull requests open for review

## 3. Allow intents to be resolved on output type (cont.)

Extends **`fdc3.findIntent()`** and **`AppMetadata`** to use the new info for resolution:

```
findIntent(intent: string, context?: Context, resultContextType?: string):
Promise<AppIntent>;
```

E.g.:

```
const appIntent = await fdc3.findIntent("ViewContact", "fdc3.ContactList");

// returns only apps that return the specified result Context type:
// {
//     intent: { name: "ViewContact", displayName: "View Contact Details" },
//     apps: { name: "MyCRM", resultContext: "fdc3.ContactList"}]
// }
```

# Pull requests open for review

## 3. Allow intents to be resolved on output type (cont.)

Does not (yet) extend **`fdc3.raiseIntent()`** to support the new **`resultContextType`** argument

- Due to the difficulty of adding a second optional string argument:

```
raiseIntent(intent: string, context: Context, app?: TargetApp):
Promise<IntentResolution>;

type TargetApp = string | AppMetadata;
```

- Note issue #506:
  Deprecate use of the string name field in favour of **`AppMetadata`**

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3

(to handle use-cases not covered by channels or intents)
(PrivateChannels and returning channels from intents)
(issue #433)(PR #508 *)

**Summary:**

There are clear use cases for being able to retrieve a channel for a context,

- E.g. A stream of prices, orders or other activity for a specified security

And requirements:

- Synchronisation & cleanup: know when listeners connect or disconnect
- Responses should come from a guaranteed source, not just any app
- Resolver support

* Incorporates previous PRs #495 and #499

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

**Changes:**

- Intent listeners should be able to return a channel (if they want to):

```
type IntentHandler = (context: Context) => Promise<IntentResult> | void;
type IntentResult = Context | Channel;
```

- Intent results can be **`Context`** or **`Channel`**:

```
export interface IntentResolution {
  readonly source: TargetApp;
  readonly version?: string;
  getResult(): Promise<IntentResult>;
}
```

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

- Update **`resultContext`** element in AppD Application schema to specify **`"channel"`** returned by an intent handler:

```
{
     "name": "Example App",
     "appId": "Example App v1.0",
     …
     "intents": [
          {
               "name": "View Orders",
               "displayName": "View Orders",
               "contexts": ["fdc3.instrument"],
               "resultContext": "channel<fdc3.order>"
          }
     ]
}
```

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

- Extends **`fdc3.findIntent()`** and **`AppMetadata`** to use the new info for resolution:

```
findIntent(intent: string, context?: Context,  resultContextType?: string):
Promise<AppIntent>;
```

E.g.:

```
const appIntent = await fdc3.findIntent("QuoteStream", instrument, "channel<fdc3.Quote>");
// returns only apps that receive specified input and stream specified result type to
channel:
// {
//     intent: { name: "QuoteStream", displayName: "Quotes stream" },
//     apps: { name: "MyOMS", resultType: "channel<fdc3.Quote>"}]
// }
```

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

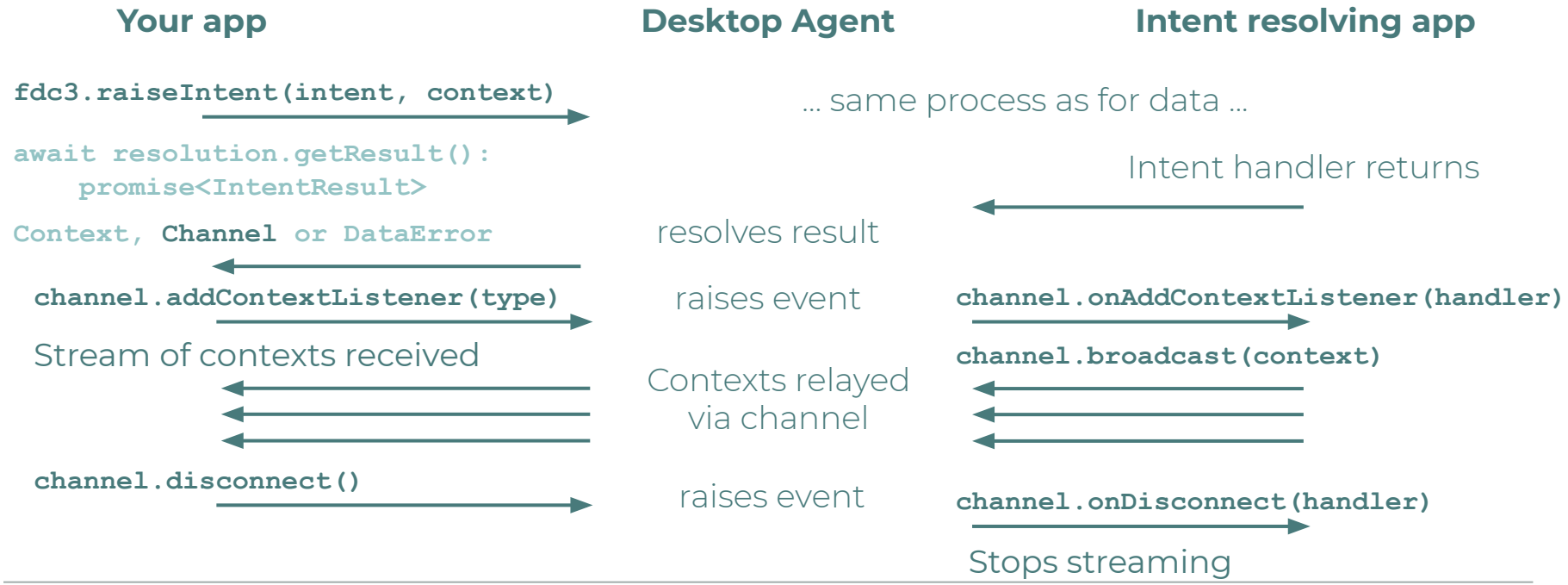
- The concept of a **`PrivateChannel`**:

  "A **`Channel`** with an auto-generated identity that is intended for private communication between applications"

  - Not discoverable and guarantees communication from only one source
  - Created via **`fdc3.createPrivateChannel()`** and can only retrieved via a raised intent (not **`fdc3.getOrCreateChannel()`**)
  - Supports event handlers for subscription and disconnect events to enable synchronisation

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)



# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

"Server-side" example:

```
fdc3.addIntentListener("QuoteStream", async (context) => {
      const channel = await fdc3.createPrivateChannel();
      const symbol = context.id.ticker;

      // add event handlers
      const addContextListener = channel.onAddContextListener((contextType) => {
            // broadcast price quotes as they come in from our quote feed
            feed.onQuote(symbol, (price) => {
                  channel.broadcast({ type: "price", price});
            });
      });
      const unsubscriberListener = channel.onUnsubscribe((contextType) => {
            feed.stop(symbol);
      });
      const disconnectListener = channel.onDisconnect(() => {
            feed.stop(symbol);
      });

      return channel;
});
```

# Pull requests open for review

## 4. Add the concept of a feed or stream to FDC3 (cont.)

"Client-side" example:

```
try {
    const res = await fdc3.raiseIntent("QuoteStream", {type:"fdc3.instrument", id: {symbol:
"AAPL"}});
    try {
        const result = await res.getResult();
        //check that we got a result and that its a channel
        if (result && result.addContextListener) {
            const listener = result.addContextListener("price", (quote) => console.log(quote));
              result.onDisconnect(() => {
                    console.warn("Quote feed went down");
              });
              // Sometime later...
              listener.unsubscribe();
        } else {
            console.warn(`${resolution3.source} did not return a channel`);
        }

     } catch(channelError){
         console.log(`Error: ${resolution3.source} returned an error: ${channelError}`);
     }
} catch (resolverError) { … }
```

# Pull requests open for review

## 5. Allow intents to be targeted at specific instances of apps (issue #450)(PR #509)

**Summary:**

There may be one or more instances of an application already open, which the user will often wish to deliver their intent and context to...

E.g.

- A user viewing details of a trade may wish to raise an intent to ViewOrder and target an open instance of a particular order management application.
- A user viewing details of an order in an OMS may wish to open the customer's details in a CRM application. The CRM app may wish to use details of the app instance that raised the intent to allow the user to ViewOrders in the instance of the app that contacted it.
- A user may wish to plot a pricing chart by raising a ViewChart intent, then add comparisons by repeatedly raising ViewChart (or ViewComparison etc.) and targeting the same instance.

# Pull requests open for review

## 5. Allow intents to be targeted at specific instances of apps (cont.)

**Changes:**

- Adds an **`instanceId`** field (+optional **`instanceMetadata`**) to **`AppMetadata`**:

```
interface AppMetadata {
    readonly name: string;
    readonly appId?: string;
    readonly version?: string;
    readonly instanceId?: string;
    readonly instanceMetadata?: Record<string, any>;
    readonly title?: string;
    readonly tooltip?: string;
    readonly description?: string;
    readonly icons?: Array<Icon>;
    readonly images?: Array<string>;
}
```

# Pull requests open for review

## 5. Allow intents to be targeted at specific instances of apps (cont.)

- Return **`AppMetadata`** from **`fdc3.open()`**:
  `open(app: TargetApp, context?: Context): Promise<AppMetadata>`
  n.b. **`AppMetadata`** is already included in an **`IntentResolution`**
- Add new **`fdc3.findInstances()`** API call:
  **`findInstances(app: TargetApp): Promise<Array<AppMetadata>>;`**
- **`AppIntent`** (findIntent) may contain multiple responses for an app with instances (+1 per instance), E.g.:

```
const appIntent = await fdc3.findIntent("StartChat");
// returns:
// { intent: { name: "StartChat", displayName: "Chat" },
//   apps: [
//    { name: "Skype" },
//    { name: "Symphony" },
//    { name: "Symphony", instanceId: "93d2fe3e-a66c-41e1-b80b-246b87120859" },
//   ]}
```

# Pull requests open for review

## 5. Allow intents to be targeted at specific instances of apps (cont.)

- Get instance metadata from another API call:

```
const instanceMetadata = await fdc3.open(appMetadata, context);

// or
const appIntent = await fdc3.findIntent("StartChat");
const instanceMetadata = appIntent.apps[2];

// or
const res = await fdc3.raiseIntent(intent, context);
const instanceMetadata = res.source;

//or
const instances = await fdc3.findInstances({name: "MyApp"});
const instanceMetadata = instances[0];
```

- Target a specific app instance via **`AppMetadata`** with an instanceId:

```
await fdc3.raiseIntent(intent, context, instanceMetadata);
```

# Pull requests open for review

## 6. Make optional: fdc3.joinChannel and fdc3.leaveChannel (issue #431)(PR #512)

**Summary:**

- fdc3.joinChannel and fdc3.leaveChannel provide programmatic access to joining and leaving channels.
- However, it is intended that the desktop agent provide UI to enable joining and leaving (system) channels, often making these functions redundant
- The spec doesn't currently limit which channels can be joined - allowing the unintended joining of 'App channels' and problems with a lack of display metadata for them (e.g. #243).
- Make these functions optional so that a Desktop Agent MAY (rather than MUST) implement them (but if they do, they MUST implement them as specified).

# Pull requests open for review

## 7. All DesktopAgent and Channel APIs should be async (issue #362)(PR #516)

**Summary:**

- The majority of DesktopAgent and Channel API calls are async, with the exception of:
  - fdc3.broadcast & channel.broadcast
  - fdc3.addIntentListener, fdc3.addContextListener & channel.addContextListener
  - fdc3.getInfo
- Make all API functions async for consistency, safety and ease of use.
- The inconsistency makes using these functions slightly different from the other API calls by:
  - adding complexity to code
  - Failing to acknowledge that most implementations of these functions will be asynchronous anyway (as they will often communicate with other windows or services in a container)
- Existing behaviour can be achieved in an async function by prefixing the calls with await

# Pull requests open for review

## 7. All DesktopAgent and Channel APIs should be async (cont.)

**Changes:**

- `fdc3.broadcast()` & `channel.broadcast()`
  - Had no return type - non-breaking change to return `promise<void>`
- `fdc3.addIntentListener()`, `fdc3.addContextListener()` & `channel.addContextListener()`
  - Change return type from `Listener` to `promise<Listener>`
  - Used only in the minority of implementations, hence, the change will not affect most implementations.
- `fdc3.getInfo()`
  - Change return type from `ImplementationMetadata` to `promise<ImplementationMetadata>`
  - new in 1.2 and usage is not yet widespread.

# Pull requests open for review

## 8. Missing ResolveError option when a user cancels raiseIntent (issue #518)(PR #522)

**Summary:**

- If a user cancels the intent resolution, there is no suitable `ResolveError` option to indicate this back to the caller of raiseIntent.
- It would be useful to indicate this scenario so the code that raises the intent can choose to accept the "error" without raising errors to the user.

**Changes:**

- Added error `UserCancelled` to the `ResolveError` enumeration to be used when the user closes the resolver UI or otherwise cancels resolution of a raised intent.

# FDC3 Channels, Feed & Transactions

Issues still open for discussion:

- Standardise auto-receipt of current context on **`joinChannel`**/**`addContextListener`** in DesktopAgent and Channel #511
- Deprecate use of the string **`name`** field in favour of **`AppMetadata`** #506

# FDC3 Channels, Feed & Transactions

Existing Documentation Improvement (mostly merged):

- [Merged] **`fdc3.joinChannel`** docs should explain expected behavior on joinChannel #418 (PR #454)
- [Merged] Clarify how apps should handle broadcast and receipt of multiple context types #434 (PR #464)
- [Merged] Documentation for **`fdc3.raiseintent`** only describes the case with a specified app #426 (PR #461)
- [Merged] Incorrect statement in FDC3 spec regarding **`fdc3.addContextListener`** being a no-op when not joined to a channel #448 (PR #449)
- [Open] Improve spec and documentation of **`fdc3.addContentListener`** to mention joining channels #451 (PR #492)