



Content hand-over

1 Contents

2	Use-cases	2
3	The API	2
4	The Manifest and schemes	2
5	When more than one app can handle a scheme or content type	3
6	Arguments	3
7	The service: Didcot	3
8	A note about content type and scheme ambiguity	4
9	See also	4

10 Content handover is when an application is asked by the user to open a file
11 it doesn't know how to handle or to start a service it is not responsible for
12 providing. For example, a browser just finished downloading an epub file and
13 the user now wants to open it. The browser will ask the system to start the
14 application responsible for dealing with that file. Another example would be
15 a map application in which the user located a destination and now wants to
16 navigate. The map app will ask the system to start the navigation app for that
17 location.

18 Use-cases

19 See [Content hand-over/Use-cases](#)¹.

20 The API

21 The SDK will have a very simple API for applications to hand-over:

```
22 void launch_app_for_uri(const char* uri)
```

23 This API will make a D-Bus call to the launcher service, Didcot. See below.

24 The Manifest and schemes

25 An application that is able to handle a given content type should declared that
26 in the manifest. Collabora recommends a section in the manifest for supported
27 content types containing a list of those supported. A different section should
28 contain the schemes the app can handle. Collabora recommends the [schemes
29 used by iOS](#)² should be adopted as much as possible for consistency and to take
30 advantage of existing knowledge.

¹https://www.apertis.org/concepts/archive/application_framework/content_hand-over_use-cases/

²https://developer.apple.com/library/ios/featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html

31 **When more than one app can handle a scheme or content** 32 **type**

33 Since content types and schemes can be handled by more than one application,
34 Didcot should be capable of providing the user with a list of options. It can be
35 similar to Android, in which the user is able to tell the system to always choose
36 a given application for a specific type or scheme.

37 The App Launcher can handle specific schemes or content types specially. For
38 instance, a scheme for getting information about a Point of Interest (POI) can
39 have a special dialog that besides showing the applications that can provide
40 details on the POI also shows a number of details itself.

41 **Arguments**

42 Any information that needs to be passed to the application should be encoded
43 in the URL. For instance, a navigation app may want an address to navigate to.
44 It might be encoded like this:

```
45 nav:30+Destination+Road?saddr=21+Source+Street
```

46 A scheme can be provided to get more information about a song. Details about
47 Shakira's Sale el Sol song could be requested like this:

```
48 media-info:Sale+el+Sol?author=Shakira&album=Waka+Waka
```

49 The arguments can be flexible, as well. The requester might be able to provide
50 information which are more or less specific depending on its own context. So
51 a query for information on a location for instance can be either a very specific
52 latitude, longitude:

```
53 location-info:Grömitz?lat=54.174730&long=10.977516
```

54 A detailed description for a place in a city:

```
55 location-info:Carat Hotel?type=hotel&city=Grömitz&country=Germany
```

56 Or even a number of stops on the way to a destination:

```
57 location-info:Grömitz?lat=54.174730&long=10.977516&stop=Bochum&lat=xx&long=yy&stop=Berlin&lat=zz&long=ww
```

58 The allowed arguments, their formats and meanings should be part of the design
59 for the specific service, and should be properly documented. Convenience APIs
60 can and should be provided for services, to make them easier to use.

61 **The service: Didcot**

62 Didcot is the service which handles launching applications. It will be responsible
63 for taking requests from applications for handling files or schemes. The service
64 will rely on the manifest information provided by the applications to know which
65 application knows how to handle a given file format or scheme. It will provide
66 a single method:

```
67 <method name="LaunchAppForURI">
68 <annotation name="org.freedesktop.DBus.GLib.Async" value=""/>
69 <arg name="uri" direction="in" type="s"/>
70 </method>
```

71 When given `file://` URIs, Didcot will sniff the content type from the file and
72 look for applications that can handle that type on the manifest database (or its
73 own cached version of it). Similarly, when given any non-file URI, Didcot will
74 look for apps that can handle that scheme. If there is more than one application
75 that can handle the file or scheme, Didcot may decide to present a dialog asking
76 the user to select which one they would prefer to use.

77 Didcot can treat certain schemes or content types as privileged and allow only
78 certain apps to handle them.

79 **A note about content type and scheme ambiguity**

80 The only situation in which Didcot should use content types is for local files.
81 A mail attachment that needs to be opened should be saved to local storage
82 and then opened like a regular file, as is done in other mobile systems such as
83 Android. Schemes such as `http` may host a great number of file types. Only
84 browsers should register for that scheme, though. If the browser cannot handle
85 a specific file format, it will download the file to local storage and open it using
86 a `file://` URI. It is tempting to try to be smarter and automatically decide that
87 a file hosted on a web site should actually be handled by a music or video player,
88 for instance. That does not work well in practice: web servers lie about file types
89 and file names lie even more. Even browsers, which are very well equipped to
90 work with that fact are sometimes fooled. The best approach is to let browsers
91 deal with web servers and let Didcot handle local files on the apps' behalf.

92 **See also**

- 93 • [Content hand-over/Design issues](https://www.apertis.org/concepts/archive/application_framework/content_hand-over_design_issues/)³.

³https://www.apertis.org/concepts/archive/application_framework/content_hand-over_design_issues/