DIAL

DIscovery **A**nd **L**aunch protocol specification

Version 1.6.4

1 Introduction

This document describes DIAL, a simple protocol for DIscovery And Launch that enables 2nd screen applications to discover and launch 1st screen applications on 1st screen devices.

The goal of this protocol is to enable CE device owners to enjoy seamless integration of phone and tablet applications with their TV-based entertainment experience.

2 Terminology

1st screen: a TV, Blu-ray player, set-top-box, or similar device
2nd screen: a smartphone, tablet, or similar device
DIAL Server: a device implementing the server side of the DIAL protocol, usually a 1st screen device.
DIAL Client: a device that can discover and launch applications on a DIAL server – usually a 2nd screen device.

3 Example use-cases

Three different methods are suggested for using DIAL and for establishing communication between 2nd screen and 1st screen apps:

- 1st screen service
- 2nd screen service
- Cloud service

3.1 1st screen service

A Netflix app on an iPhone discovers a Netflix-enabled TV and then launches the Netflix app on the TV to watch a movie. Steps (a) and (b) below are covered by DIAL, steps (c) and (d) are Netflix-specific and outside the scope of this document:

(a) Netflix app on iPhone discovers DIAL service on the networked TV
(b) Netflix app on iPhone uses DIAL to ask TV to launch Netflix app
(c) Netflix app on iPhone discovers Netflix app on TV
(d) Netflix app on iPhone and Netflix app on TV communicate to show the movie

3.2 2nd screen service

A YouTube app on an Android tablet discovers a YouTube-enabled TV and then launches the YouTube app on the TV and plays a video on the TV; after the video ends, the TV returns to the previously playing TV show. Steps (a) and (b) below are covered by DIAL, step (c) is YouTube-specific and outside the scope of this document:

(a) YouTube app on tablet discovers DIAL service on the networked TV
(b) YouTube app on tablet uses DIAL to ask TV to launch YouTube app, passing an IP/port corresponding to the YouTube app on the tablet
(c) YouTube app on TV communicates with YouTube app on tablet to show the video, then exits back to normal TV UI

3.3 Cloud service

A (fictitious) WebcamX app on an Android phone discovers a WebcamX-enabled TV and then launches the browser-based HTML5 WebcamX app on the TV to display a webcam stream. Steps (a) and (b) below are covered by DIAL, step (c) is WebcamX-specific and outside the scope of this document:

(a) WebcamX app on phone discovers DIAL service on the networked TV
(b) WebcamX app on phone uses DIAL to ask TV to launch an HTML5-based browser starting with the URL for the WebcamX app, passing a unique token based on a random number
(c) WebcamX app on phone communicates with WebcamX app on TV via a cloud based WebcamX server, using the unique token to enable the server to correctly route traffic between the two apps

4 Protocol overview

The DIAL protocol has two components, **DIAL Service Discovery** and the **DIAL REST Service**.

DIAL Service Discovery enables a DIAL client device to discover DIAL servers on its local network segment and obtain access to the DIAL REST Service on those devices.

The *DIAL REST Service* enables a DIAL client to query, launch and optionally stop applications on a DIAL Server device.

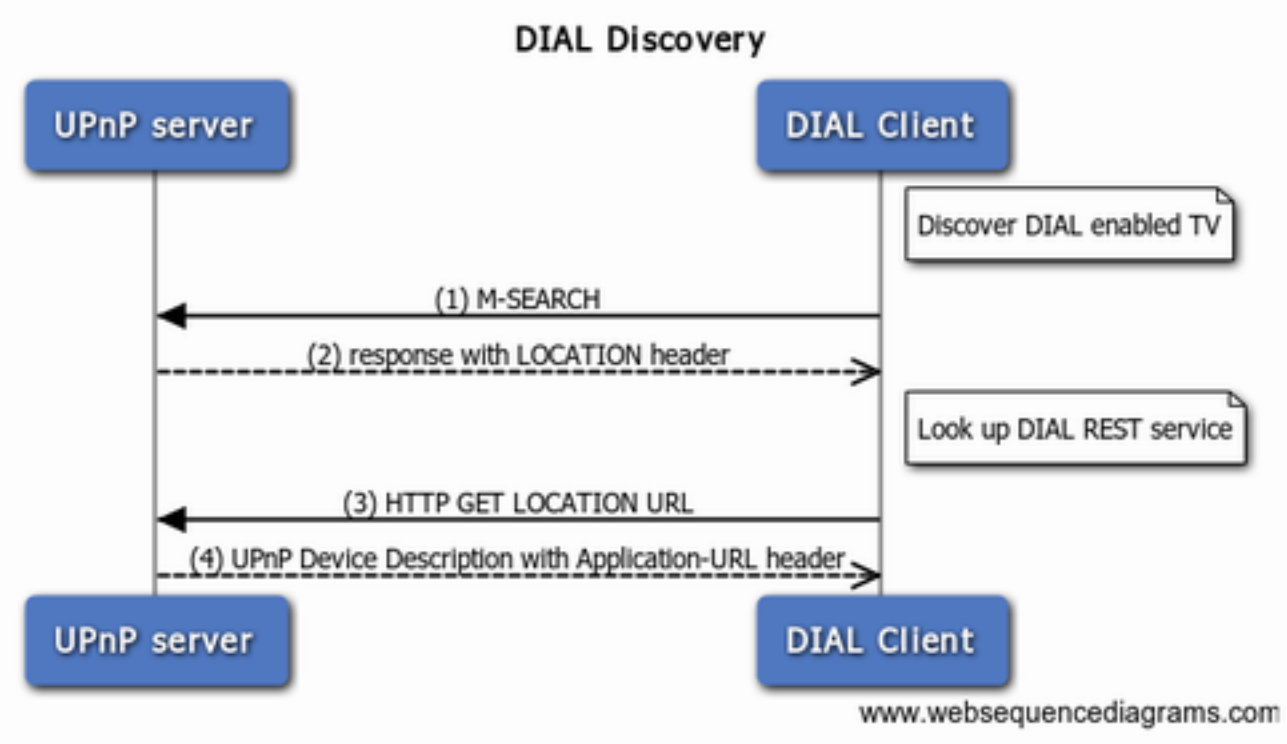
DIAL Service Discovery is achieved using a new Search Target within the SSDP protocol defined by UPnP [1] and an additional header in the response to an HTTP request for the UPnP device description.

The DIAL REST Service is accessed using HTTP [2].

DIAL clients and servers SHALL support the requirements of Section 2.1 of [1]. In particular, HTTP 1.0 SHALL be supported and support of HTTP 1.1 is RECOMMENDED.

5 DIAL Service Discovery

The DIAL Service Discovery protocol is based on SSDP (Simple Service Discovery Protocol) version 1.1 as defined in UPnP [1] and HTTP [2] and is illustrated in the following figure:



5.1 M-SEARCH request

A DIAL client that wishes to discover DIAL servers SHALL send an `M-SEARCH` request as defined in Section 1.3.2 of [1] over UDP to the IPv4 multicast address 239.255.255.250 and UDP port 1900 including the Search Target header (`ST`) with the following value defined by this specification:

```
urn:dial-multiscreen-org:service:dial:1
```

5.2 M-SEARCH response

An SSDP/UPnP server receiving an `M-SEARCH` request with the Search Target defined above shall respond as defined in Section 1.3.3 of [1], including a `LOCATION` header containing an absolute HTTP URL for the UPnP description of the root device. The host portion of the URL SHALL either resolve to an IPv4 address or be an IPv4 address.

The Search Target header (`ST`) of the response SHALL contain the identifier defined in Section 5.1.

5.3 Device description request

On receipt of the M-SEARCH response, the DIAL client shall issue an HTTP GET request to the URL received in the `LOCATION` header of the M-SEARCH response. (*Note that matching of SSDP header field names is case-insensitive*).

5.4 Device description response

On receipt of a valid HTTP GET for the device description, a DIAL server SHALL respond with an HTTP response containing the UPnP device description as defined in Section 2 of [1].

In addition to the requirements of [1], the request SHALL NOT be redirected.

If the request is successful, the HTTP response SHALL contain an additional header field, `Application-URL`, the value of which SHALL be an absolute HTTP URL. This URL, minus any trailing backslash ('/') character, identifies the DIAL REST Service and is referred to as the *DIAL REST Service URL*. The host portion of the URL SHALL either resolve to an IPv4 address or be an IPv4 address.

The format of the Application-URL header field is defined as follows, following the ABNF notation of [2]:

```
Application-URL = "Application-URL" ":" absoluteURI
```

A DIAL client receiving this response SHALL use the provided URL to access the DIAL REST service defined below. (*Note that matching of HTTP header names is case-insensitive*)

Implementation Note:

- *The UPnP friendlyName field of the device description may be useful for presentation in the DIAL client device UI, for example when offering users a choice of DIAL server devices to interact with.*

6 DIAL REST Service

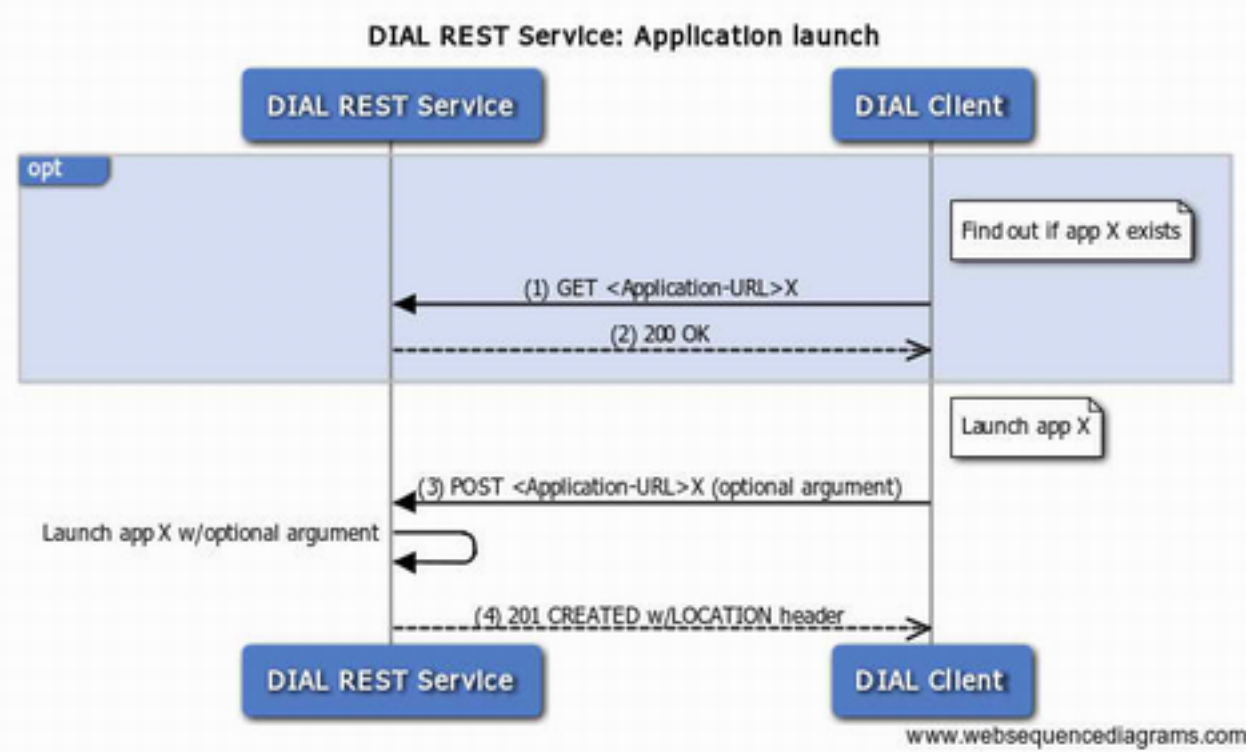
6.1 Application Resources

The DIAL REST service represents applications (for example Netflix, YouTube, etc.) as resources identified by URLs. Operations related to an application are performed by issuing HTTP requests against the URL for that application, known as an *Application Resource URL*.

The Application Resource URL for an application is constructed by concatenating the *DIAL REST Service URL,* a single backslash character *('/')* and the *Application Name*.

The Application Name for each application is defined by the application provider. Application Names MUST be registered in the DIAL Registry (See Section 7).

6.1.1 Launching an Application



6.1.1.1 Client Request

A DIAL client that wishes to launch an application on a DIAL server SHALL send an HTTP `POST` request to the Application Resource URL for the desired application.

Implementation Note:

- *Which applications are available to launch using DIAL is entirely at the discretion of the DIAL Server implementer.*

The message body of the POST request MAY be empty or MAY contain an argument string, to be passed to the application on launch.

If the message body of the POST request is empty, a " Content-Length: 0" header SHOULD be sent to avoid receiving a possible "411 Length Required" error.

If the message body of the POST request is non-empty, the MIME type SHALL be `text/plain`, the character encoding SHALL be UTF-8 and the character encoding SHALL be indicated explicitly by including the `charset` MIME parameter.

Implementation Notes:

- *How the argument is passed to the application may be platform- and app-specific and is outside the scope of this document.*
- *The format of the argument should match the requirements of the application being launched. Key-value pairs, JSON, XML, etc. are all possible and the choice is outside the scope of this document.*
- *DIAL servers are required to support argument strings up to 4KB in length.*
- *A DIAL server MAY pass the argument string directly to the application: Applications MUST NOT assume that any security checks (such as character encoding checks) have been performed and therefore MUST perform their own security checks on the argument data.*
    - *The DIAL payload MUST be passed to the launched application in a way that ensures it **cannot be used to circumvent fundamental security or integrity of the platform**. For example, the DIAL payload SHOULD NOT be used to*
        - *directly define arbitrary command-line parameters when running a native application unless adequate protections are taken.*
        - *provide the actual executable name.*
        - *control process attributes like priority or ownership*

6.1.1.2 Server response

On receipt of a valid `POST` request, the DIAL Server SHALL first extract the Application Name from the `POST` URL. If the `POST` request is invalid or cannot be processed to extract the Application Name, the server SHALL return the appropriate HTTP response code as defined in [2].

The server SHALL as indicated in the table below, depending on:

- whether the Application Name is not recognized,
- whether the content length of the message body exceeds the maximum size supported by
- the server
- the current application state:
    - not started: the application is not running
    - starting: the application is being started due to another DIAL REST Service request or for another reason
    - running: the application is running
- whether the message body is empty or non-empty

DIAL servers MUST support any messages body in a POST requests with content length 4KB or less.

In the table below, earlier rows take precedence over later ones. If the Action indicates an HTTP response code the server SHALL return an HTTP response with the indicated response code.

Application	Message body	Application state	Action

recognized			
No	any	n/a	`404 Not Found`
Yes	too long	n/a	`413 Request Entity Too Large`
Yes	empty	Not running	Start application
Yes	non-empty	Not running	Start application with provided argument, if any
Yes	empty	Starting[1]	No action
Yes	non-empty	Starting	No action
Yes	empty	Running	No action
Yes	non-empty	Running	Provide new argument to application

If the application is running after the action specified above, the DIAL Server SHALL return an HTTP response with response code `201 Created`. In this case the `LOCATION` header of the response shall contain an absolute HTTP URL identifying the running instance of the application, known as the *Application Instance URL*. The host portion of the URL SHALL either resolve to an IPv4 address or be an IPv4 address. No response body shall be returned.

Otherwise, i.e. if the application cannot be successfully started or re-started for any reason, the DIAL Server SHALL return an HTTP response with response code `503 Service Unavailable`.

Implementation Note:

- *If the application is already running and an argument is provided, and the platform does not support providing new arguments to running applications, then the application should not be restarted with the new argument unless allowed by the application provider.*
- *To run the application, the host system should use a non-blocking form of "fork", "exec", "spawn", or equivalent that will catch any immediate error (like ENOMEM or EAGAIN) and map it to a 503 "Service Unavailable" error. If there is no error and the system command to run the application succeeds, then the application is running for the purposes of this specification.*

The Application Instance URL may be used to request information about and to stop the running instance of the application, as described below.

6.1.2 Stopping an application

6.1.2.1 Client request

A DIAL client that wishes to stop a running instance of an application on a DIAL server SHALL send an HTTP `DELETE` request to the Application Instance URL.

[1] Started by DIAL or by any other mean, e.g. built-in menu, etc...

6.1.2.2 Server response

Support of the `DELETE` request is RECOMMENDED for DIAL Servers.

If the `DELETE` request is not supported the DIAL server SHALL return an HTTP response with response code `501 NOT IMPLEMENTED`.

If supported, then on receipt of a `DELETE` request, the DIAL Server SHALL first determine whether the provided URL corresponds to a running application. If the `DELETE` request is invalid or cannot be processed to determine whether the provided URL corresponds to a running application, the server SHALL return the appropriate HTTP response code as defined in [2].

If the provided URL does not correspond to an application running in the foreground, the server SHALL return an HTTP response with response code `404 Not Found`.

Otherwise, the server SHALL send an HTTP response with response code `200 OK` and attempt to stop the running application. These two operations SHOULD be carried out asynchronously.

6.1.3 Querying for application information

6.1.3.1 Client request

A DIAL client that wishes to discover information about an application SHALL send an HTTP `GET` request to the Application Resource URL.

6.1.3.2 Server response

On receipt of a `GET` request, the DIAL Server SHALL first extract the Application Name from the request URL. If the `GET` request is invalid or cannot be processed to extract the Application Name, the server SHALL return the appropriate HTTP response code as defined in [2].

If the Application Name is not recognized, the server SHALL return an HTTP response with response code `404 Not Found`.

Otherwise the DIAL server SHALL return an HTTP response with response code `200 OK`. The MIME type of the response SHALL be `text/xml` and the character encoding SHALL be UTF-8 and SHALL be explicitly indicated with the `charset` MIME parameter.

The XML document SHALL conform to the schema defined in Annex A except that unrecognised XML elements and attributes MUST be ignored by the client.. The semantics of this document are described in the following table:

Element or @attribute	Definition
`name`	Contains the Application Name
`options`	

`@allowStop`	If true, indicates that the `DELETE` operation described in Section 6.1.2 is supported. If false, indicates that the `DELETE` operation decribed in Section 6.1.2 is not supported.
`state`	`running` indicates that the application is installed, starting or running `stopped` indicates that the application is installed and not running A string beginning `installable=` indicates that the application is not installed but is available for installation Any other value is invalid and SHOULD be ignored.
`link`	Optional element that SHOULD be included *when an application is running*. The exception is when stopping an application is not supported, <link> doesn't have to be provided. The value of the `rel` attribute MUST be "run". The `href` attribute contains the resource name of the running application, e.g. "**run**" or "**pid-25352**". This name SHOULD match the last portion of the name returned in the 201 CREATED response (e.g. http://192.168.1.200:2342/apps/YouTube/**run**)

DIAL Servers MAY support client triggering of application installation for specific applications which are not currently installed.

If the Application Name is recognized, the application is not installed and the DIAL server supports client triggering of application installation for this application, then the DIAL Server SHALL return a `state` element beginning with the string `installable=`. The remainder of the element MUST contain an absolute HTTP URL. The host portion of the URL SHALL either resolve to an IPv4 address or be an IPv4 address.

A DIAL client may initiate installation of the application on the DIAL server by sending an HTTP GET request to the provided URL.

On receipt of such a request at the DIAL Server, if the Application Name is recognized, the application is not installed and the DIAL server does not support triggering of application installation for this Application Name, then the DIAL server SHALL return an HTTP response with response code `404 Not Found`.

Implementation note:

- *If an app is installable, the client can choose whether or not to GET the installable URL provided. For example, the client (2nd screen app) can present a choice to the user: "Application is not installed. Would you like to install it now?". The way the 1st screen device handles the installation request is OEM-defined; options include immediate installation of the app, presentation of the app in a marketplace or similar app store, etc.*

7 DIAL Registry

To ensure that the correct name for each application is well-defined, and to avoid naming conflicts, Application Names must be registered in the DIAL Registry. The DIAL specification maintainer will maintain this registry and make it available to anyone implementing or using the DIAL protocol.

Application Names may be registered explicitly, or a set of Application Names with a common prefix may be registered by registering an *Application Prefix*.

Application Names and Prefixes MUST consist of a sequence of characters matching the `pchar` production of RFC3986 [3]. Matching of Application Names and Prefixes is case-sensitive and SHALL be performed after decoding percent-encoded characters (i.e. the three character sequence '%30' matches the single character '0'.)

An Application Prefix MUST be at least four (4) characters in length (after decoding percent-encoded characters) and must include a recognizable company name. For example: "Acme-", "com.acme".

Application names can be registered if the application is actually available in the market and there is no conflict with previously registered names or prefixes. Application prefixes can be registered if the company name in the prefix is actively delivering applications or devices that run them to the market. Names or prefixes that may be confused with previous registrations (e.g. "netflix", "youtube", "Y0uTube", etc.), or are not intended for use with DIAL, may not be registered.

8 References

[1] UPnP™ Device Architecture 1.1 , 15 October 2008, http://upnp.org/sdcps-and-certification/standards/device-architecture-documents/
[2] RFC2616 (http://www.ietf.org/rfc/rfc2616.txt)
[3] RFC3986 (http://tools.ietf.org/html/rfc3986)

9 Acknowledgements

This document greatly benefited from review, feedback, and suggestions from several CE manufacturers and content providers. The document authors appreciate their help and support.

Notably, Samsung and Sony provided significant guidance to ensure that DIAL would be a compatible and effective solution for 1st screen devices and also meet their goals for great 2nd screen user experiences. The document authors especially thank our colleagues at these companies for their efforts on behalf of DIAL.

Annex A: Application resource XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:dial-multiscreen-org:schemas:dial"
        attributeFormDefault="unqualified"
        elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="urn:dial-multiscreen-org:schemas:dial"
       xmlns:atom="http://www.w3.org/2005/Atom">
   <xs:import namespace="http://www.w3.org/2005/Atom" schemaLocation="atom.xsd"/>
   <xs:element name="service" type="ServiceType"/>

   <xs:complexType name="ServiceType">
      <xs:sequence>
         <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
         <xs:element name="options" type="OptionsType" minOccurs="0" maxOccurs="1"/>
         <xs:element name="state" type="xs:string" minOccurs="1" maxOccurs="1"/>
         <xs:element name="link" type="atom:LinkType" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="OptionsType">
      <xs:attribute name="allowStop" type="xs:boolean" use="optional"/>
   </xs:complexType>
</xs:schema>
```

Annex B: Message Examples

B.1 M-SEARCH

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: urn:dial-multiscreen-org:service:dial:1
USER-AGENT: OS/version product/version
```

B.2 M-SEARCH response

```
HTTP/1.1 200 OK
LOCATION: http://192.168.1.1:52235/dd.xml
CACHE-CONTROL: max-age=1800
EXT:
BOOTID.UPNP.ORG: 1
SERVER: OS/version UPnP/1.1 product/version
ST: urn:dial-multiscreen-org:service:dial:1
```

B.3 Device description request

Message sent to 192.168.1.1, port 52235:

```
GET /dd.xml HTTP/1.1
...
```

B.4 Device description response

```
HTTP/1.1 200 OK
Application-URL: http://192.168.1.1:12345/apps
...

<UPnP device description in message body>
```

B.5 Application launch request

Message sent to 192.168.1.1, port 12345:

```
POST /apps/YouTube
Content-Type: text/plain; charset="utf-8"
...
```

```
param1=value1&param2=value2
```

B.6 Application launch response

```
HTTP/1.1 201 CREATED
LOCATION: http://192.168.1.1:12345/apps/YouTube/run
...
```

B.7 Application Information Request

Message sent to 192.168.1.1, port 12345:

```
GET /apps/YouTube
...
```

B.8 Application Information response

```
HTTP/1.1 200 OK
…

<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="urn:dial-multiscreen-org:schemas:dial">
    <name>YouTube</name>
    <options allowStop="true"/>
    <state>running</state>
    <link rel="run" href="run"/>
</service>
```

Annex C: Sequence diagram source

```
title DIAL Discovery
participant "UPnP server" as U
participant "DIAL Server" as S
participant "DIAL Client" as C
note right of C: Discover DIAL enabled TV
C->S: (1) M-SEARCH
S-->C: (2) response with LOCATION header
note right of C: Look up DIAL REST service
C->U: (3) HTTP GET LOCATION URL
U-->C: (4) UPnP Device Description with Application-URL header

title DIAL REST Service: Application launch
participant "DIAL REST Service" as S
participant "DIAL Client" as C
opt
note right of C: Find out if app X exists
C->S: (1) GET <Application-URL>X
S-->C: (2) 200 OK
end
note right of C: Launch app X
C->S: (3) POST <Application-URL>X (optional argument)
S->S: Launch app X w/optional argument
S-->C: (4) 201 CREATED w/LOCATION header
```

CHANGE HISTORY

Version 1.6.4

1. Title: Cleaned up title page.
2. 6.1.1.1: Added recommendation to send a “Content-Length: 0” header when POST is empty to avoid receiving a possible “411 Length Required” error.

Version 1.6.3

1. 6.1: Simplified the explanations of the concatenation rules for the application name with Application-URL.
2. 6.1.1.1: Added emphasis on security measures when handing off DIAL payload to application in implementation notes.
3. 6.1.1.2: Clarified we are talking about the LOCATION header.
4. 6.1.1.2: Implementation notes: Clarified that restarting an application due to a modified DIAL payload is only allowed if an application provider allows it.
5. 6.1.3.2: Clarified that an application that has just launched (is starting) should be reported as “running”.
6. Added CHANGE HISTORY section to document.