

Introduction

Documentation »
ACCESS Linux Platform Native Development »
Exchange Manager »
Introduction

The ACCESS Linux Platform Exchange Manager is a general purpose mechanism for announcing and utilizing services provided by applications, libraries, and servers. This makes it an ideal mechanism to support the ACCESS Linux Platform activity model.

What is the Exchange Manager? [^TOP^](#)

The ACCESS Linux Platform Exchange Manager does more than simply pass data back and forth. The Exchange Manager is a central system broker that facilitates communication between different applications or servers. The Exchange Manager allows data and any kind of service to be exchanged between clients and handlers:

- Client: An application requesting a service
- Handler: An application, library, or server that performs the service requested by the client

The Exchange Manager is used by applications (handlers) to inform the system of what types of services they can provide. For example, a phone application can dial a phone number, or a contacts application can store vCard data as a contact entry.

When a requesting application (client) sends a request to the Exchange Manager for a particular service, the Exchange Manager determines which handler can perform a request of that type. The Exchange Manager then invokes the appropriate application in transient mode. After the handler performs the requested service, the transient application is dismissed, and the user is returned to the calling application.

The Exchange Manager operates via a daemon and a set of APIs to perform the following functions:

- Register services
- Enable a client application to request a service
- Dispatch the service request to the appropriate handler

The handler can be on either the local device or on a remote device or server that is connected to the local device.

How Does the Exchange Manager Work? [^TOP^](#)

Client applications do not specify which application should provide the requested service. Rather, clients request services without knowing which application will process that service. The Exchange Manager is responsible for responding to the request by directing it to the appropriate handler, and by passing back any results to the client.

The advantages of not specifying a specific application to handle a request include:

- One handler can easily be replaced by another without requiring changes to the client application
- Multiple applications can register to handle a given action class

This makes the ACCESS Linux Platform activity model supported by the Exchange Manager extremely flexible and powerful: as the user adds new applications to their device, existing applications can gain new or improved capabilities.

Structuring Requests ^TOP^

The combination of verb and subject defines a specific service to the Exchange Manager:

- Verb: An action, such as "display"
- Subject: The item on which the verb acts, such as "image/jpeg"

Requests can optionally contain data objects. Each data object is defined by a name, a size, a data type, and possibly a description, if desired.

To customize the service's behavior, string or integer parameters can be associated with the service. For example, consider a request to "store" a "vObject." To specify the actual object to be stored, you can attach a parameter to the service request that provides the name of the object.

Client applications can find services in two ways:

- Statically by specifying the appropriate verb and subject
- Dynamically by asking the Exchange Manager

When finding a service dynamically, a client can ask a general question like: "What can I do with data of this_type?" Alternatively, if the client application wants to perform a specific action, such as "send," it can ask: "How can I send data of this_type?"

ACCESS Linux Platform defines a number of services that any application can use through the Exchange Manager. Therefore, most services are requested dynamically and use existing verbs that correspond to a symbolic action.

However, when a specific service is necessary, the request must include the exact verb and optional parameters.

Chapter 3, "Exchange Manager Handlers," lists those services and explains what they do and how they can be customized by using certain parameters. The services are organized by action classes, which are described in the following section.

Understanding Action Classes and Verbs ^{^TOP^}

An action class defines a symbolic action, such as Send, Get, or Import. A method is a way to perform that action. For example, to send an object as an email attachment, you would specify action class "Send" with the method "asEmailAttach".

An application builds a request for service independently of the chosen verb, as methods are often selected by the user. In other words, users can opt to send objects in a number of ways: by email attachment, by MMS, by Bluetooth, etc. For this reason, it is necessary that all methods for a given action class use the same conventions for parameters and data passing. Unless specified otherwise, the default conventions are to require zero mandatory request parameters and to pass or return data using data objects.

Application developers can define customization parameters to use in their handler, but these parameters must be optional; the requesting application may blindly invoke the handler without knowing which parameters are required. However, if the client application specifically invokes a handler, it may be useful to pass additional information to customize the default behavior.

{moscomment}