LMS(Lightning Message Serice)

Introduced in the Winter ’20 release by Salesforce, Lightning Message Service is the out-of-the-box functionality that empowers you to communicate between Visualforce and Lightning Components, including Aura web components (AWC) and Lightning web components (LWC).

LMS is defined as the standard publish-subscribe library that enables communication quickly and seamlessly with DOM across the Salesforce UI tech stack, including Visualforce Pages, Aura components, and Lightning Web Components (LWC) using simple API. With this unique feature of Salesforce, you can publish messages and subscribe to them anywhere within the lightning page. Lightning Message Service feature is similar to Aura application events that enable seamless communication between components.

Lightning Message Service is based on Lightning Message Channels, a new metadata type. Lightning Message Channel is used to access the Lightning Message Service API and Lightning Message Channel in LWC via scoped module @salesforce/messageChannel. When it comes to Visualforce, you can use the global variable $MessageChannel. In Aura, you can use lightning:messageChannel in your component.

Uses Of Lightning Message Service

1. To enable communication between Visualforce page, Lightning web components, and Aura components,
2. To access Lightning Message Service API for publishing messages throughout Lightning Experience. Also, it helps in subscribing to messages that originated from anywhere within Lightning Experience via Lightning Message Channel.
3. A specific namespace is associated with Lightning Message Channel. Developers can choose whether they want their message channel to be available to other namespaces or not, ensuring the security of communication on Lightning Message Service.

Benefits Of Lightning Message Service

4. One of the significant benefits is increased development time. For instance, if a Visualforce Page or Lightning component tries to reference a message channel that is non-available and that message channel is not exposed, then the code would not compile.
5. This messaging service offers referential integrity between the code that references them and the message channel. It restricts the deletion of message channels, which are referenced by other codes. Further, Lightning Message Service supports auto-adding message channels to packages.
6. As the metadata type is packageable, you can associate a message channel to a particular namespace and make it available/unavailable to other namespaces.

How To Use Lightning Message Service?

Lightning Message Service is based on a new type of metadata called Lightning Message Channels, which are lightweight, packageable components that you can create in your organization during runtime for publishing and subscribing to messages on them.

Structure For Lightning Message Channel

```
<?xml version=”1.0″ encoding=”UTF-8″?>

<LightningMessageChannel xmlns=”http://soap.sforce.com/2006/04/metadata”>

   <description>This is a sample Lightning Message Channel.</description>

   <isExposed>true</isExposed>

   <lightningMessageFields>

      <description>This is the data</description>

      <fieldName>data</fieldName>

   </lightningMessageFields>

   <masterLabel>SampleMessageChannel</masterLabel>

</LightningMessageChannel>
```

7. masterLabel is the only required field that identifies the message channel in various UIs.
8. Exposed is the optional boolean field that declares false if you don’t specify it. Also, this field tells our system whether or not a specific message channel is available to components in other namespaces.

Where Do We Make A Lightning Message channel In Our Org?

9. Use VsCode, then use the Salesforce CLI and SFDX project.
10. Create a folder “messageChannels” in “force-app\main\default\.”
11. In “messageChanel” directory <channelName>.messageChannel-meta.xml E.g., SampleMessageChannel.messageChannel-meta.xml.
12. Deploy the message channel to your organization to know about your message channel and then use it in your code.
13. Message channel in another namespace that can be used with the namespace followed by two underscores. So if SampleMessageChannel was from the example namespace, you’d say example__SampleMessageChannel__c.

Pubsub Model−Publish Subscribe Model

The Publish-Subscribe pattern/model is the same as the Application Event in Lighting Component. If you want to communicate between components that are not bound in parent-child relationship, but available in the same page (say, in the same app-builder page), then we will use the publish-subscribe pattern.

Here Salesforce already publishes a pubsub.js file which we will use.