Paragon (/)

paragon.messagebus

Use paragon.messagebus API to enable apps to send and receive messages through the Paragon Messaging Service.

Summary

Events
onConnected
onDisconnected
onError
onMessage
Methods
publishMessage
sendMessage
subscribe
unsubscribe
getState

Events

onConnected

Fired when connected.

addListener

```
paragon.messagebus.onConnected.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onDisconnected

Fired when disconnected.

addListener

```
paragon.messagebus.onDisconnected.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this:

		`function() {...};`

onError

Fired when an error is encountered.

addListener

```
paragon.messagebus.onError.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onMessage

Fired when a message is encountered.

addListener

```
paragon.messagebus.onMessage.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

Methods

publishMessage

Publish a message to all subscribers of a topic

```
paragon.messagebus.publishMessage(string topic, object message, function callback)
```

Parameters		
string	topic	Topic to which the message is to be published
object	message	Content to be published.
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

sendMessage

Send a message to one of the subscribers (to a topic) in a round-robin fashion. Use publish if intent is to deliver to all subscribers

```
paragon.messagebus.sendMessage(string topic, object message, string responseId, function ca
llback)
```

Parameters		
string	topic	Specific topic that the subscriber is subscribed to
object	message	Content to be sent.
string	responseId	responseId is a client side application provided string value to coordinate the receipt of a response if the other application listening for the messages on the topic decides to respond. It is optional
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

subscribe

Subscribe to the topic referenced.

```
paragon.messagebus.subscribe(string topic, string responseId, function callback)
```

Parameters		
string	topic	Topic corresponding to the message
string	responseId	responseId to coordinate the receipt of a response
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

unsubscribe

Unsubscribe from the topic referenced.

```
paragon.messagebus.unsubscribe(string topic, string responseId, function callback)
```

Parameters		
string	topic	Topic corresponding to the message
string	responseId	responseId to coordinate the receipt of a response
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

getState

Gets the state of the socket

```
paragon.messagebus.getState(string topic, function callback)
```

Parameters		
string	topic	Topic corresponding to the message
function	callback	The *callback* parameter should be a function that looks like this: `function(string state) {...};`