Service Bus

ServiceBus Queues

ServiceBus Queues are an alternative to Storage Queues that might be useful in scenarios where more advanced messaging features are needed (larger message sizes, message ordering, single-operaiton destructive reads, scheduled delivery) using push-style delivery (using long polling).

The service can use Shared Access Signature authentication, or ACS authentication.

Service bus namespaces created using the Azure portal after August 2014 no longer support ACS authentication. You can create ACS compatible namespaces with the Azure SDK.

Shared Access Signature Authentication

To use Shared Access Signature authentication, create the service bus service with:

```
from azure.servicebus import ServiceBusService

key_name = 'RootManageSharedAccessKey' # SharedAccessKeyName from Azure portal
key_value = '' # SharedAccessKey from Azure portal
sbs = ServiceBusService(service_namespace,
                        shared_access_key_name=key_name,
                        shared_access_key_value=key_value)
```

ACS Authentication

To use ACS authentication, create the service bus service with:

```
from azure.servicebus import ServiceBusService

account_key = '' # DEFAULT KEY from Azure portal
issuer = 'owner' # DEFAULT ISSUER from Azure portal
sbs = ServiceBusService(service_namespace,
                        account_key=account_key,
                        issuer=issuer)
```

Sending and Receiving Messages

The **create_queue** method can be used to ensure a queue exists:

```
sbs.create_queue('taskqueue')
```

The **send_queue_message** method can then be called to insert the message into the queue:

```
from azure.servicebus import Message

msg = Message('Hello World!')
sbs.send_queue_message('taskqueue', msg)
```

It is then possible to call the **receive_queue_message** method to dequeue the message.

```
msg = sbs.receive_queue_message('taskqueue')
```

ServiceBus Topics

ServiceBus topics are an abstraction on top of ServiceBus Queues that make pub/sub scenarios easy to implement.

The **create_topic** method can be used to create a server-side topic:

```
sbs.create_topic('taskdiscussion')
```

The **send_topic_message** method can be used to send a message to a topic:

```
from azure.servicebus import Message

msg = Message('Hello World!')
sbs.send_topic_message('taskdiscussion', msg)
```

A client can then create a subscription and start consuming messages by calling the **create_subscription** method followed by the **receive_subscription_message** method. Please note that any messages sent before the subscription is created will not be received.

```
from azure.servicebus import Message

sbs.create_subscription('taskdiscussion', 'client1')
msg = Message('Hello World!')
sbs.send_topic_message('taskdiscussion', msg)
msg = sbs.receive_subscription_message('taskdiscussion', 'client1')
```

Event Hub

Event Hubs enable the collection of event streams at high throughput, from a diverse set of devices and services.

The **create_event_hub** method can be used to create an event hub:

```
sbs.create_event_hub('myhub')
```

To send an event:

```
sbs.send_event('myhub', '{ "DeviceId":"dev-01", "Temperature":"37.0" }')
```

The event content is the event message or JSON-encoded string that contains multiple messages.