IRIDIUM CLOUDCONNECT

MARCH 2020

JACO BOTES, SR MGR IoT PRODUCT MANAGEMENT

iridium®

COMPREHENSIVE SATELLITE IoT PORTFOLIO

Customers

marine instruments · Onix SMART2 · MOBILITEX · SkyBitz · ZUNIBAL · Wireless innovation · MetOcean · AST · pivotel. · MARLINK · CATERPILLAR® · HITACHI · Kkomatsu · HYUNDAI · GARMIN · NAL Research Corporation · Honeywell · TRACK 24

Channel

VAR | SP | OEM | Consumer / Personnel

Products

Narrowband Finished Products | IoT Transceivers | Iridium Certus IoT Products | Third Party VAM products

IOT Services

Iridium Certus℠ IoT (Message Based) | Iridium Certus™ IP | Iridium CloudConnect | aws

Network Services

SBD® | RUDICS | Iridium Certus IoT (4.8 – 100Kbps) | GMDSS | LPGAN | IoT Extend | LoRa®

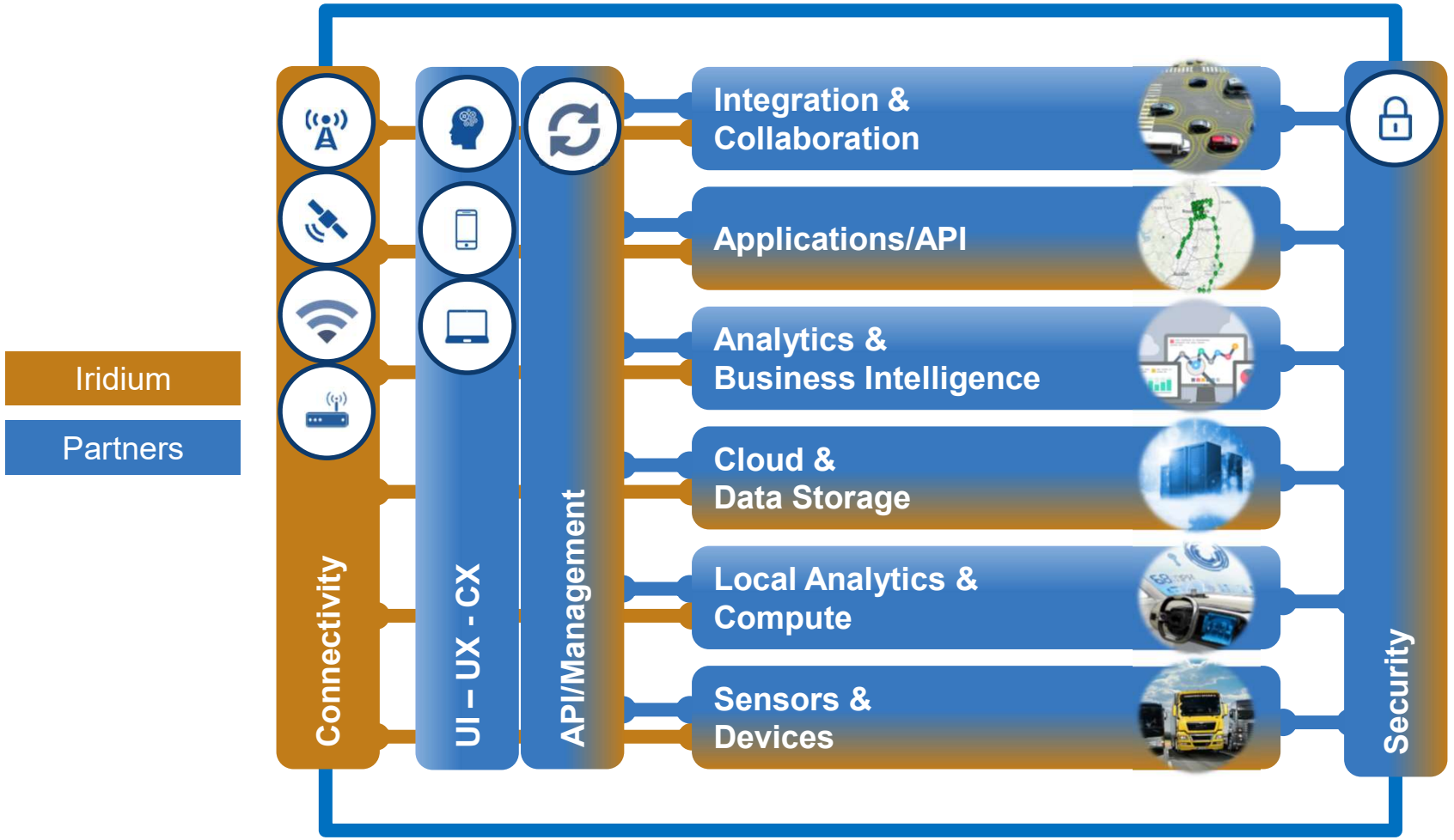
Networks

Iridium® Constellation | Partner Networks

Iridium Existing Network Based Products | Iridium Network Based Products in development | Third Party Networks

IRIDIUM AND IoT PARTNER ECOSYSTEM

IoT covers the entire ecosystem of critical information traversing between components of an end-to-end system. It includes considerations for connectivity, data storage, data analytics and the various ways in which data is consumed, shared and secured.

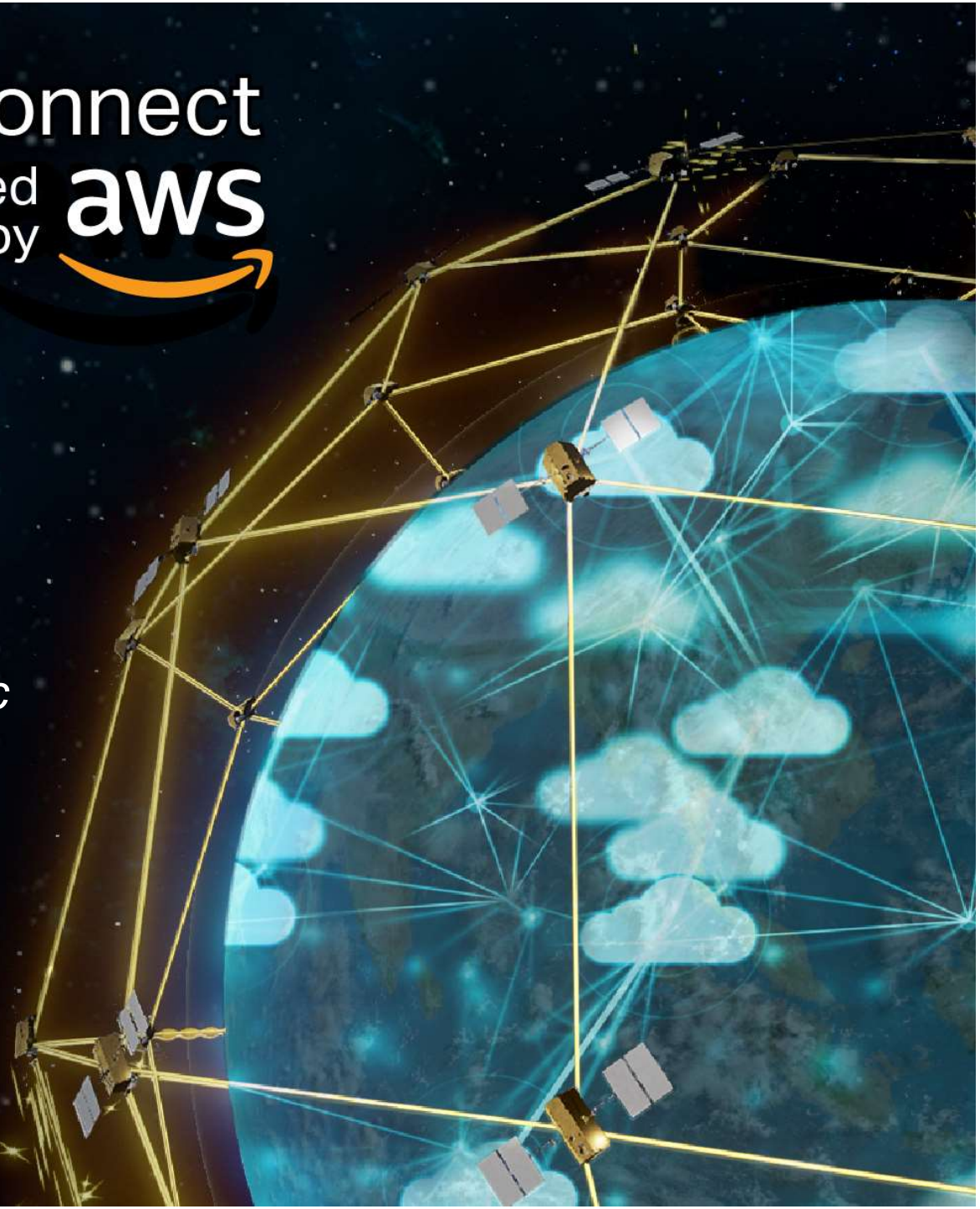


Iridium CloudConnect

powered by aws

A value-added service co-developed by Iridium and Amazon Web Services (AWS).

It provides a means for IoT and messaging traffic to be transferred directly to cloud instances where customers are storing data.



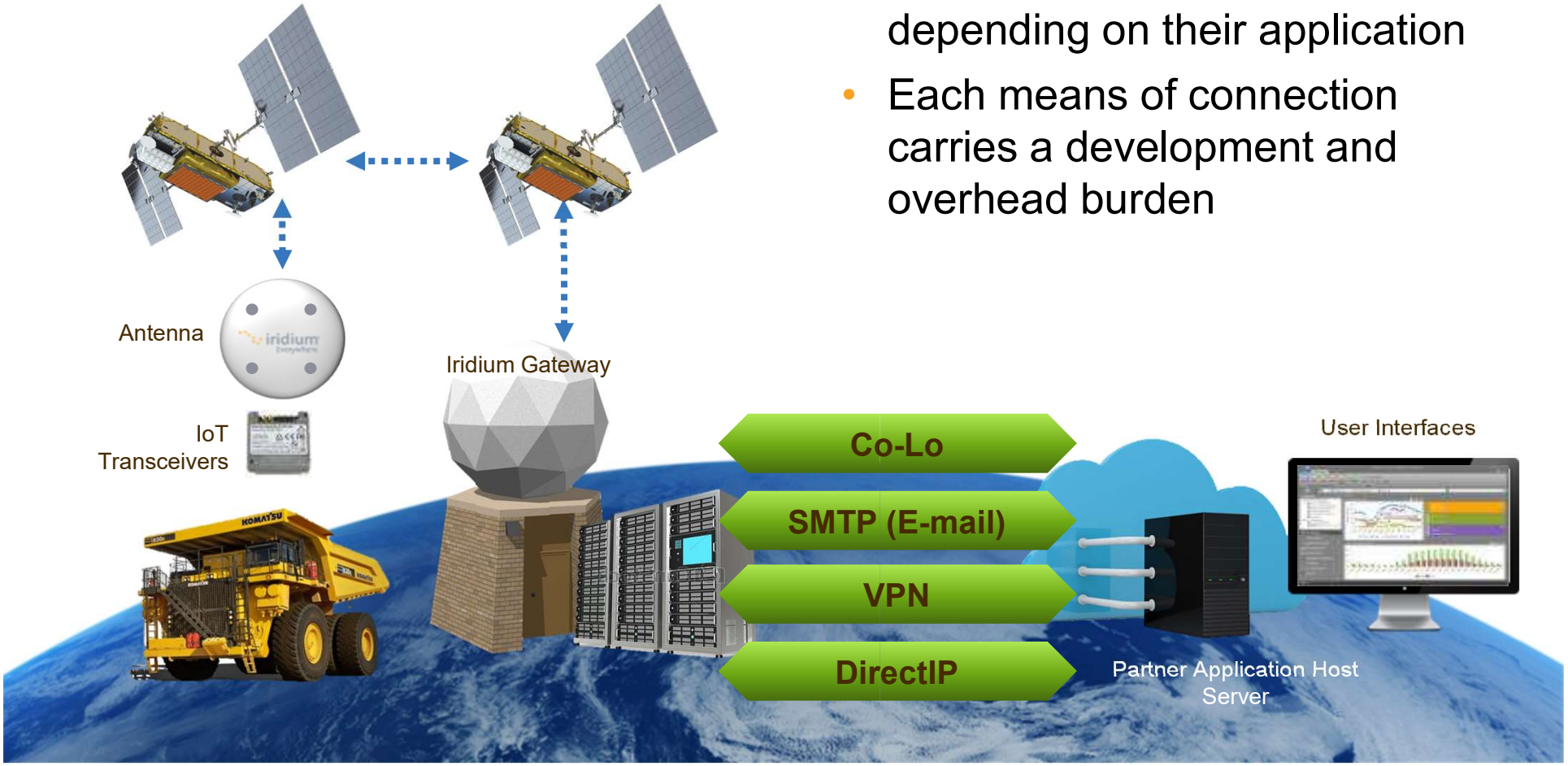
iridium®

IRIDIUM CLOUDCONNECT / OVERVIEW

- Value added service co-developed by Iridium and AWS
- Provides a means for Iridium IoT messages to be transferred directly to cloud instances
- Cloud is now the de-facto way for businesses to deliver IT applications and services.

- Iridium CloudConnect is an adapter service that transfers SBD data from am Iridium-based device directly to a specified cloud service, initially AWS.
- SBD® data sent from a device through the Iridium CloudConnect service remains through closed carrier networks and dedicated private lines to Iridium and AWS, allowing you to utilize lightweight protocols between your device and your IoT backend.

CONNECTING TO IRIDIUM FOR SBD® DATA



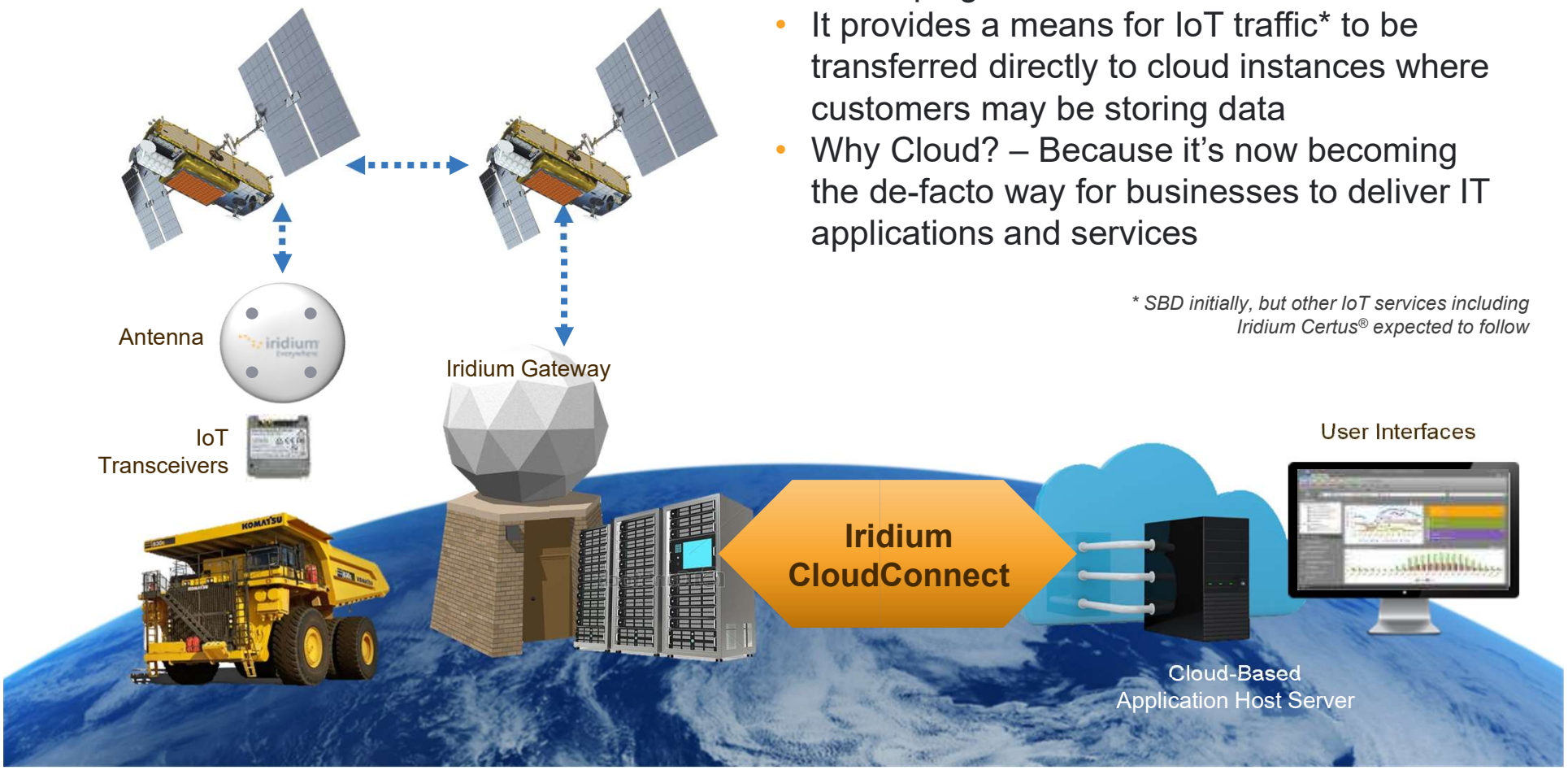
- Customers connect to Iridium to deliver/extract SBD data, depending on their application
- Each means of connection carries a development and overhead burden

IRIDIUM CLOUDCONNECT

powered by aws

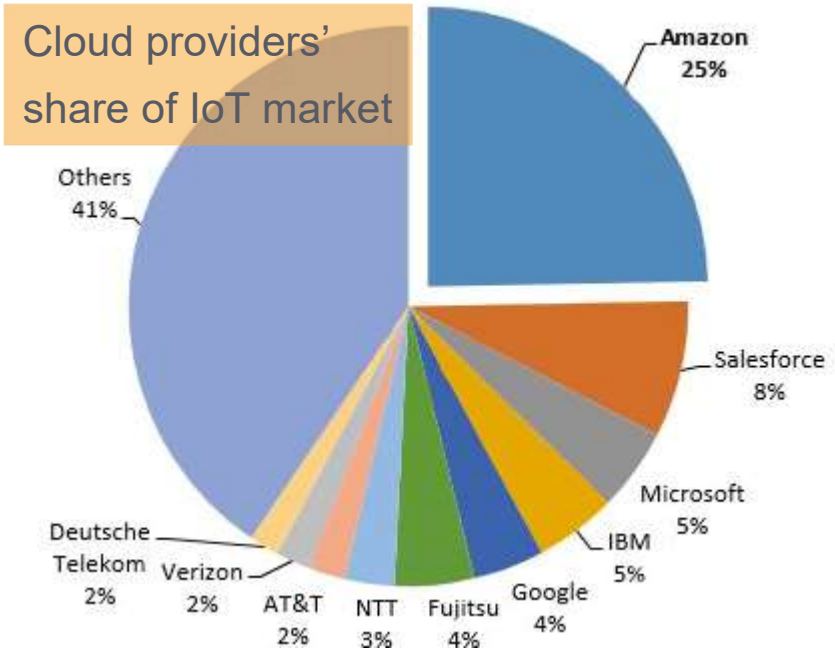
- Iridium CloudConnect is a value-added service that Iridium and AWS are co-developing.
- It provides a means for IoT traffic* to be transferred directly to cloud instances where customers may be storing data
- Why Cloud? – Because it's now becoming the de-facto way for businesses to deliver IT applications and services

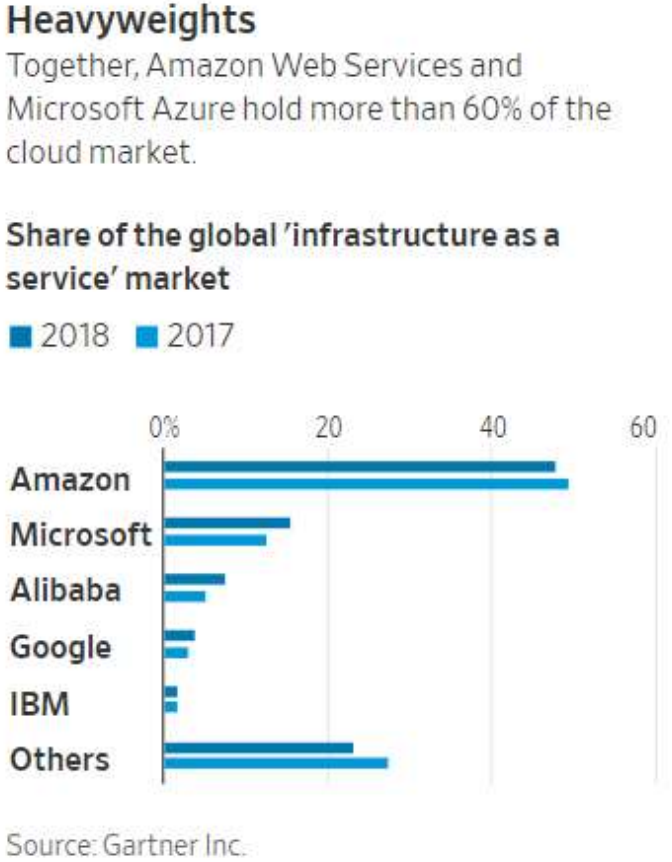
** SBD initially, but other IoT services including Iridium Certus® expected to follow*



iridium

AWS FIRST





- High IoT Security Standards
- Serverless Architecture
- Powerful AWS IoT Analytics Paired With AI and Machine Learning
- AWS Has a Strong Partner Network of IoT Device Manufacturers
- Integration Across a Sheer Number of AWS Products and Services

AWS is our preferred initial go-to-market partner but can be expanded to others

BUSINESS OUTCOMES WITH IoT

Revenue growth

IoT data drives business growth

Operational Efficiency

IoT data decreases OpEx

powered by aws

IoT SOLUTIONS ARE COMPLEX & MULTIDIMENSIONAL

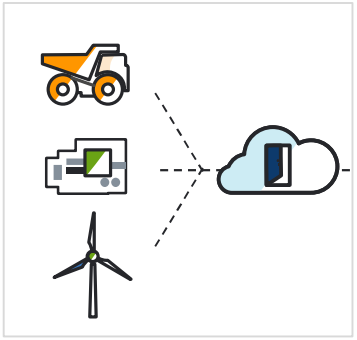
Onboard, provision, manage	Connect, communicate, secure	Analyze, visualize, act	Engage, empower, delight	Transform, shift culture

powered by aws

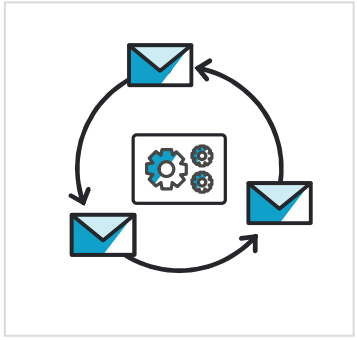
AWS IoT CORE

Secure Device Connectivity and Messaging

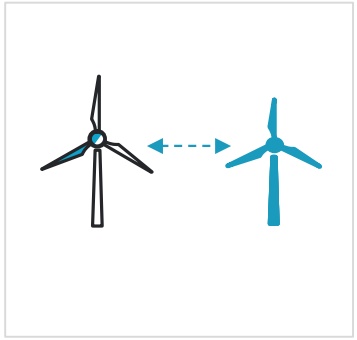
AWS IoT Core is a managed service that lets connected devices easily and securely interact with cloud applications and other devices.



To securely connect devices to the AWS cloud and other devices at scale



To route, process, and act upon data from connected devices

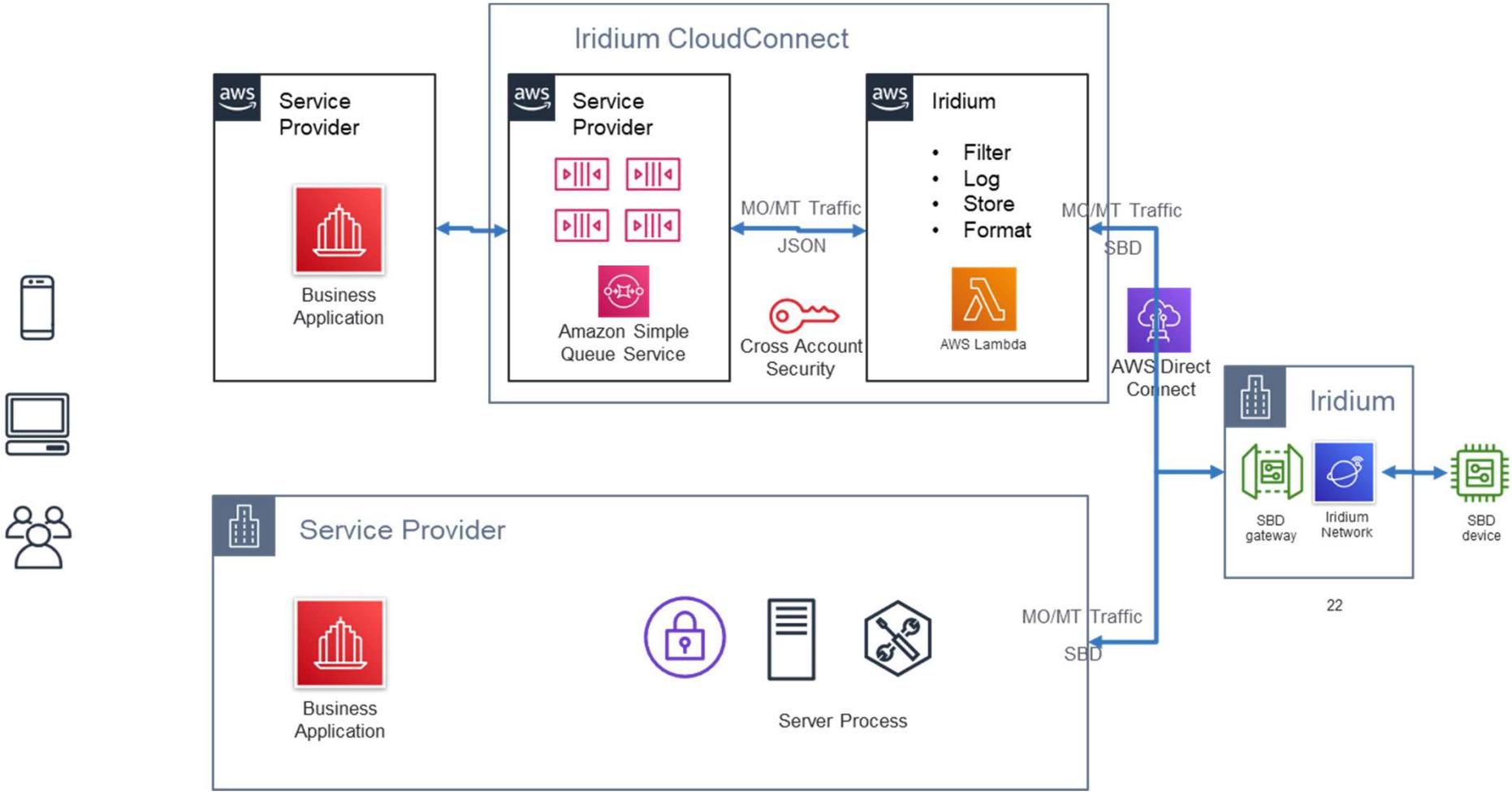


To enable applications to interact with devices even when they are offline

To fully integrate with other AWS service to reason on top of the data (Analytics, Databases, AI, etc.)

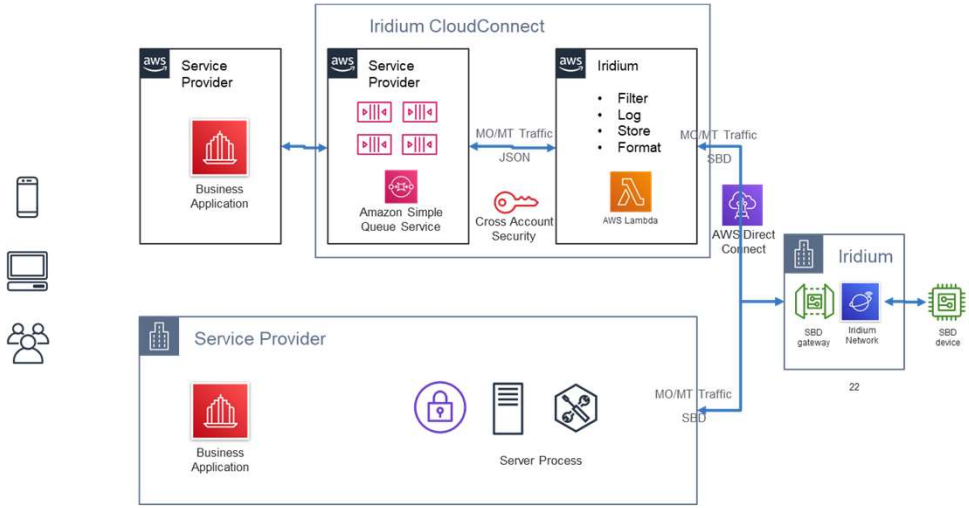
powered by aws

CUSTOMER SOLUTION / IRIDIUM CLOUDCONNECT



CHARACTERISTICS & BENEFITS

- Simplicity for our Partners.
- Leverages AWS global, redundant, infrastructure and support.
- Iridium installed a redundant direct connection to the AWS infrastructure.
- SBD messages converted to JSON are placed in a Simple Queue Service (SQS) queue in the Partners AWS cloud environment.
- Iridium will add/upgrade future messaging options without changes to the Iridium CloudConnect Partner interface.
- Does not replace any SBD functionality - current backend delivery options remain as they are today.



Iridium CloudConnect is aligned with the design principles for AWS Well Architected Foundation solutions:

- Operational Excellence
- Security, Reliability
- Performance Efficiency
- Cost Optimization

SBD® ←→ JSON

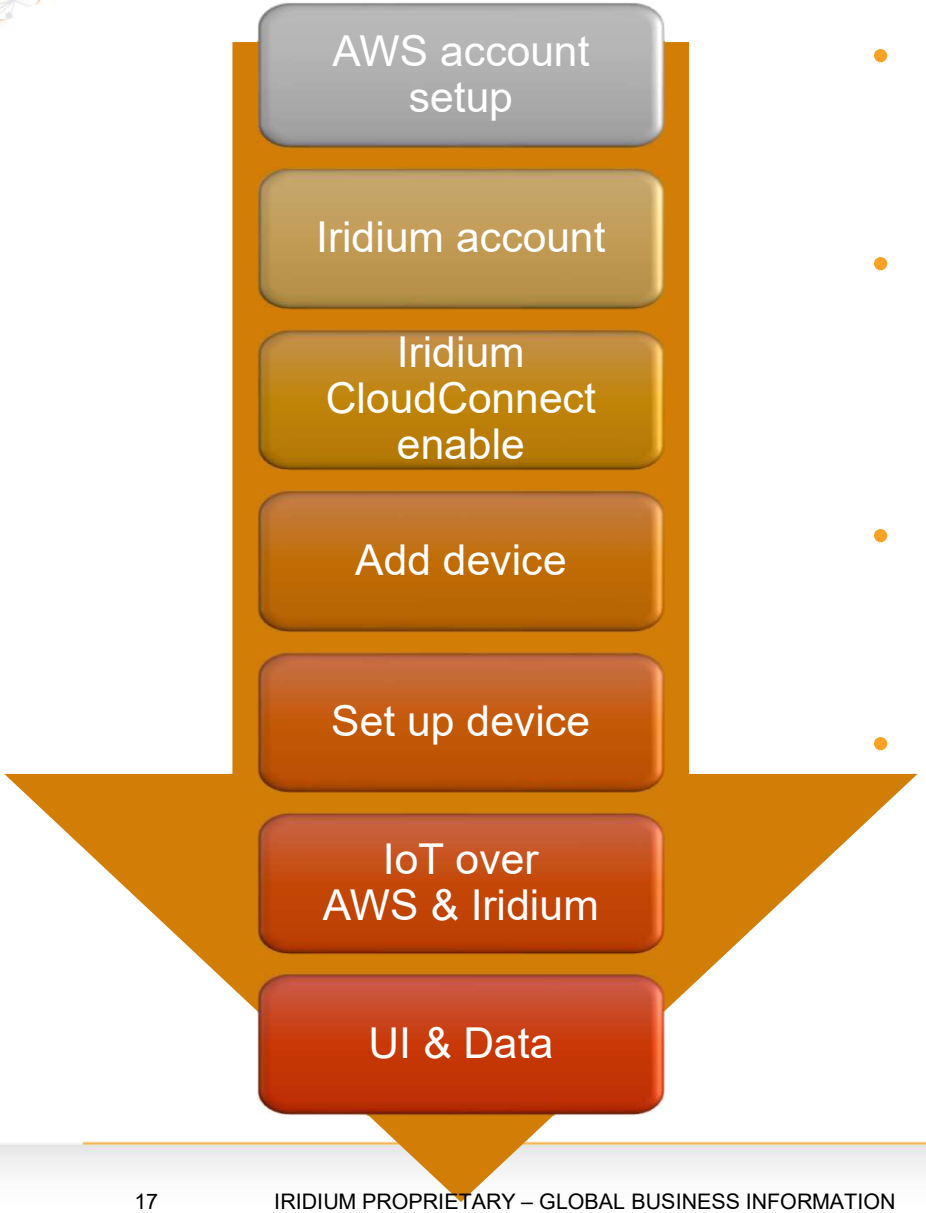
```
0100 3d01 001c 6c69 0d6c 3330 3032 3334
3036 3430 3734 3236 3000 0089 0000 5c70
57df 0300 0b01 2b91 f954 4f64 0000 0003
0200 0d01 004e
```


```
{
    "api_version": 1,
    "data":
    {
    "mo_header":
        {
        "cdr_reference": 1179650258,
        "session_status_int": 0,
        "session_status": "No error.",
        "momsn": 58939,
        "mtmsn": 0,
        "imei": "300334010407160",
        "time_of_session": "2019-12-16 15:04:09"
        },
    "location_information":
        {
        "cep_radius": 10,
        "latitude": "38.52137",
        "longitude": "-77.12970"
        },
    "payload": "746573746d756963"
    }
}
```

IRIDIUM CLOUDCONNECT COMPARED TO DIRECTIP

- Both provide similar functionality
  - SBD MO delivered to the Iridium SBD gateway to a specific IP endpoint
  - Send SBD MT data from your IP endpoint to the device via the SBD gateway and satellite constellation
- DirectIP as a more general purpose allows SBD data exchange to a general IP end point and requires you to translate the SBD messages into your own format for processing and use.
- Iridium CloudConnect allows you to exchange messages in JSON format for both MO and MT messages and includes cloud configuration to generate SQS queues to support Iridium CloudConnect in your AWS instance.

STEPS TO GET IRIDIUM “CLOUDCONNECTED”



- Iridium and partner set up cross-account security role using AWS console (IAM user)
- Iridium provides CloudFormation template to Partner (creates all the AWS infrastructure in the Partner AWS EC2 instance)
- SBD devices set to use Iridium CloudConnect IP as destination (SPnet)
- Partner extracts/insert MO/MT from queues in their own AWS account instead of via the SBD gateway (DirectIP/Email/VPN)

CLOUDFORMATION INTEGRATION

Queue Options

- CloudFormation script that creates FIFO SQS queues (preferred)
- CloudFormation script that creates standard (non-FIFO) SQS queues
- CloudFormation script that sets up access without creating SQS queues (customer will use their own queues)

Creates

- The MO SQS queue where MO messages will be deposited.
- The MT SQS queue where MT messages will be deposited.
- The MTConfirmation SQS queue where MT confirmation messages will be deposited.
- The MTErrors SQS queue where errors in MT delivery will be deposited.
- A cross-account IAM role that allows Iridium access to the above SQS queues.

INTERACTING WITH SQS MO EXAMPLE


```
{
    "api_version": 1,
    "data":
    {
    "mo_header":
        {
        "cdr_reference": 1179650258,
        "session_status_int": 0,
        "session_status": "No error.",
        "momsn": 58939,
        "mtmsn": 0,
        "imei": "300334010407160",
        "time_of_session": "2019-12-16 15:04:09"
        },
    "location_information":
        {
        "cep_radius": 10,
        "latitude": "38.52137",
        "longitude": "-77.12970"
        },
    "payload": "746573746d756963"
    }
}
```

INTERACTING WITH SQS – MO EXAMPLE

```
# Get the service resource
sqs = boto3.resource('sqs')
# Get the queue
queue = sqs.get_queue_by_name(QueueName='ICCMO.fifo')
# Process messages by printing out IMEI, location, and payload
for message in queue.receive_messages(WaitTimeSeconds=30):
header = None
location = None
payload = None
data = json.loads(message.body)
for iei in data['data']
if data['iei_type'] == 'MO Header IEI' header = data
elif data['iei_type'] == 'MO Location Information IEI' location = data
elif data['iei_type'] == 'MO Payload IEI' payload = data
# The message may contain location information
if location is not None:
print('IMEI {0} location is {1}'.format(
header['imei'],
location['latitude_longitude'] ))
# The message may contain a payload
if payload is not None:
print('IMEI {0} payload of {1} bytes'.format(
header['imei'],
payload['mo_payload'] ))
# Let the queue know that the message is processed
message.delete()
```

INTERACTING WITH SQS – MT EXAMPLE

Bare Minimum

```
{
      "client_message_id" : 1234,
      "message" : "68656c6c6f20776f726c64",
      "imei" : "300234087352917“
}
```

With Priority Specified

```
{
      "client_message_id" : 9977331,
      "message" : "68656c6c6f20776f726c64",
      "imei" : "300234087352917",
      "priority" : 2
}
```

All possible options
(see SBD dev guide for allowed combinations)

```
{
      "client_message_id" : 789012,
      "message" : "68656c6c6f20776f726c64",
      "imei" : "300234087352917",
      "flush_mt_queue" : false,
      "send_ring_alert_no_payload" : false,
      "message_priority" : 3,
      "assign_mtmsn" : false
}
```

INTERACTING WITH SQS – MT EXAMPLE

```
import boto3
import json
import uuid

# Get the service resource
sqs = boto3.resource('sqs')

# Get the queues
mt_queue = sqs.get_queue_by_name(QueueName = 'ICCMT.fifo')

message = "hello device".encode("utf-8").hex()

# Create a message
body = {
    "client_message_id": 123456789,
    "message": message,
    "imei": "432143214321432"
}

deduplication_id = str(uuid.uuid4())
group_id = str(uuid.uuid4())

mt_queue.send_message(
    MessageBody = json.dumps(body),
    MessageGroupId = group_id,
    MessageDeduplicationId = deduplication_id
)
```

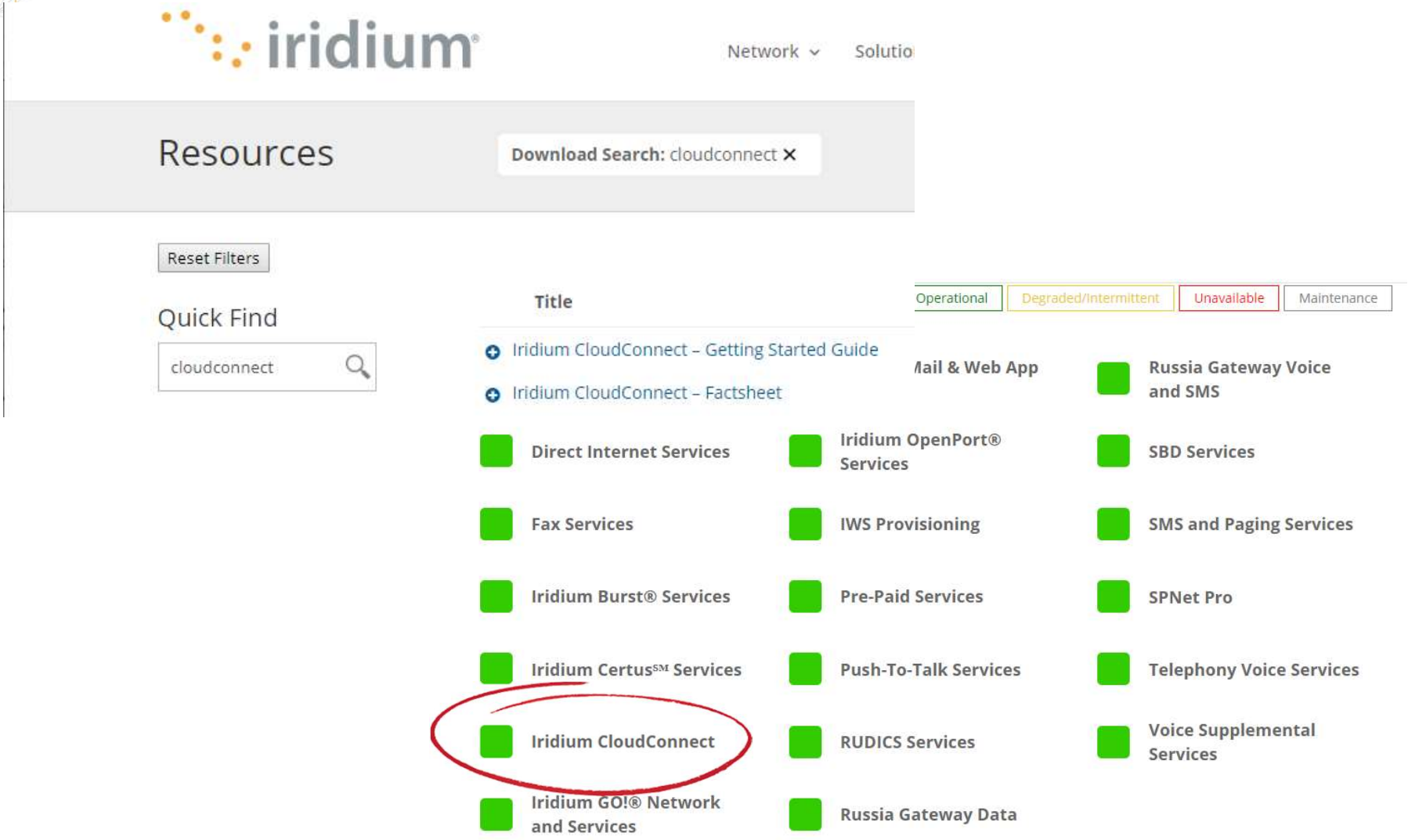
INTERACTING WITH SQS – MT CONFIRMATION QUEUE EXAMPLE

```
# Get the service resource
sqs = boto3.resource('sqs') # Get the queues
mt_confirmation_queue =
sqs.get_queue_by_name(QueueName='ICCMTConfirmation.fifo')
# Check the confirmation queue and, if no message found, check the error
queue
for message in mt_confirmation_queue.receive_messages(WaitTimeSeconds=30):
data = json.loads(message.body)
# Status greater than zero indicates success
if data['mt_message_status'] > 0:
print('Message {0} is queued for delivery to IMEI {1} in position
{2}'.format( data['unique_client_message_id'],
data['imei'],
data['mt_message_status']
)) else:
print('Message {0} was not sent to IMEI {0}'.format(
data['unique_client_message_id'],
data['imei']
))
# Let the queue know that the message is processed
message.delete()
```

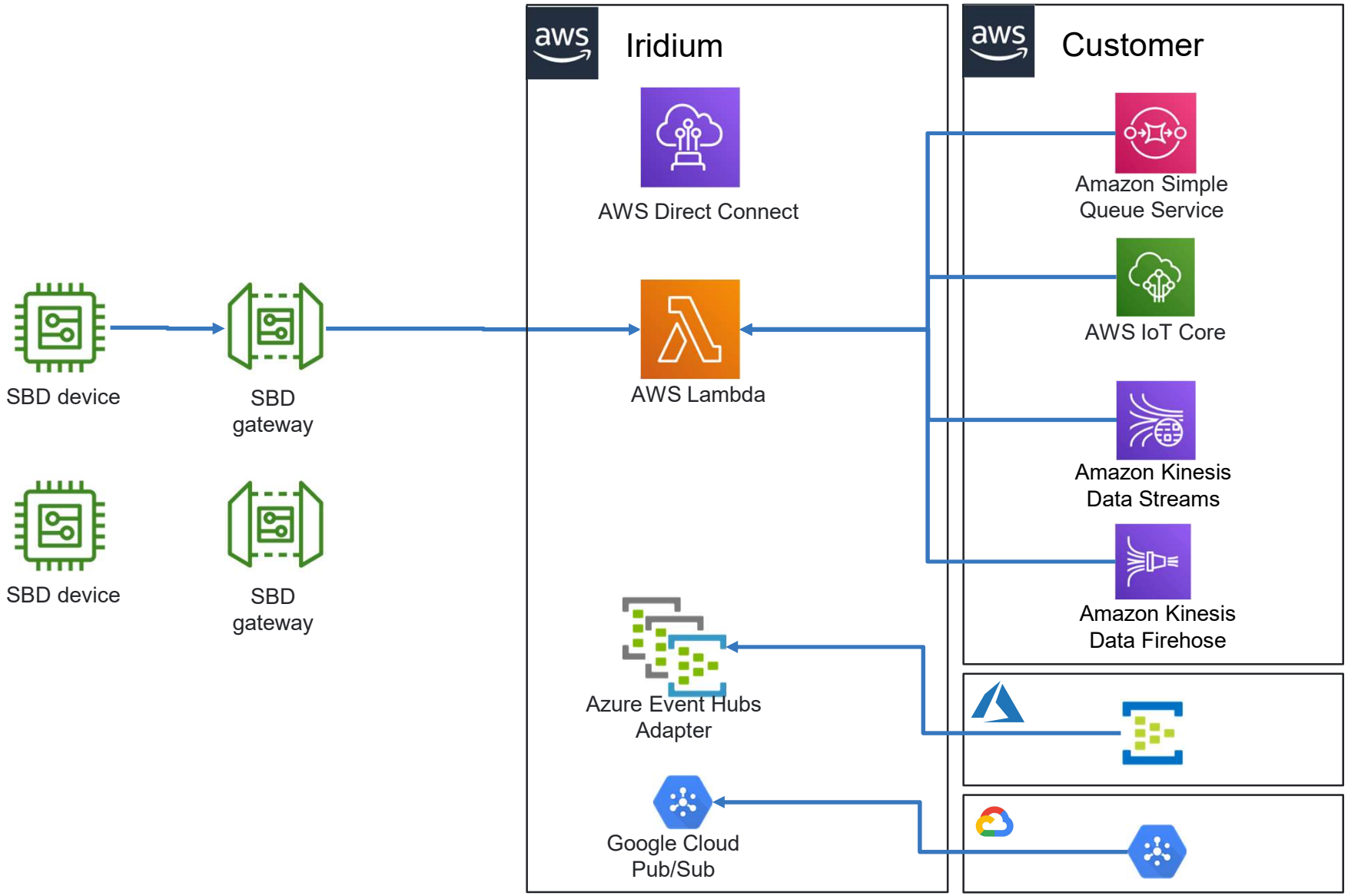
INTERACTING WITH SQS – MT ERROR QUEUE EXAMPLE

```
# Get the service resource
sqs = boto3.resource('sqs')
# Get the queues
mt_errors_queue = sqs.get_queue_by_name(QueueName='ICCMTErrors.fifo')
# Check the confirmation queue and, if no message found, check the error queue
for message in mt_errors_queue.receive_messages(WaitTimeSeconds=30):
print(message.body)
# Let the queue know that the message is processed
message.delete()
```

IRIDIUM FOR PARTNERS & RESOURCES



IRIDIUM CLOUDCONNECT



ROADMAP ITEMS

2019:

- SBD MO/MT to customer AWS SQS
- Beta completed 2019
- Service live EOY 2019
- Significant testing and verification on multiple scenarios.
- DirectConnect to AWS

2020: Consideration

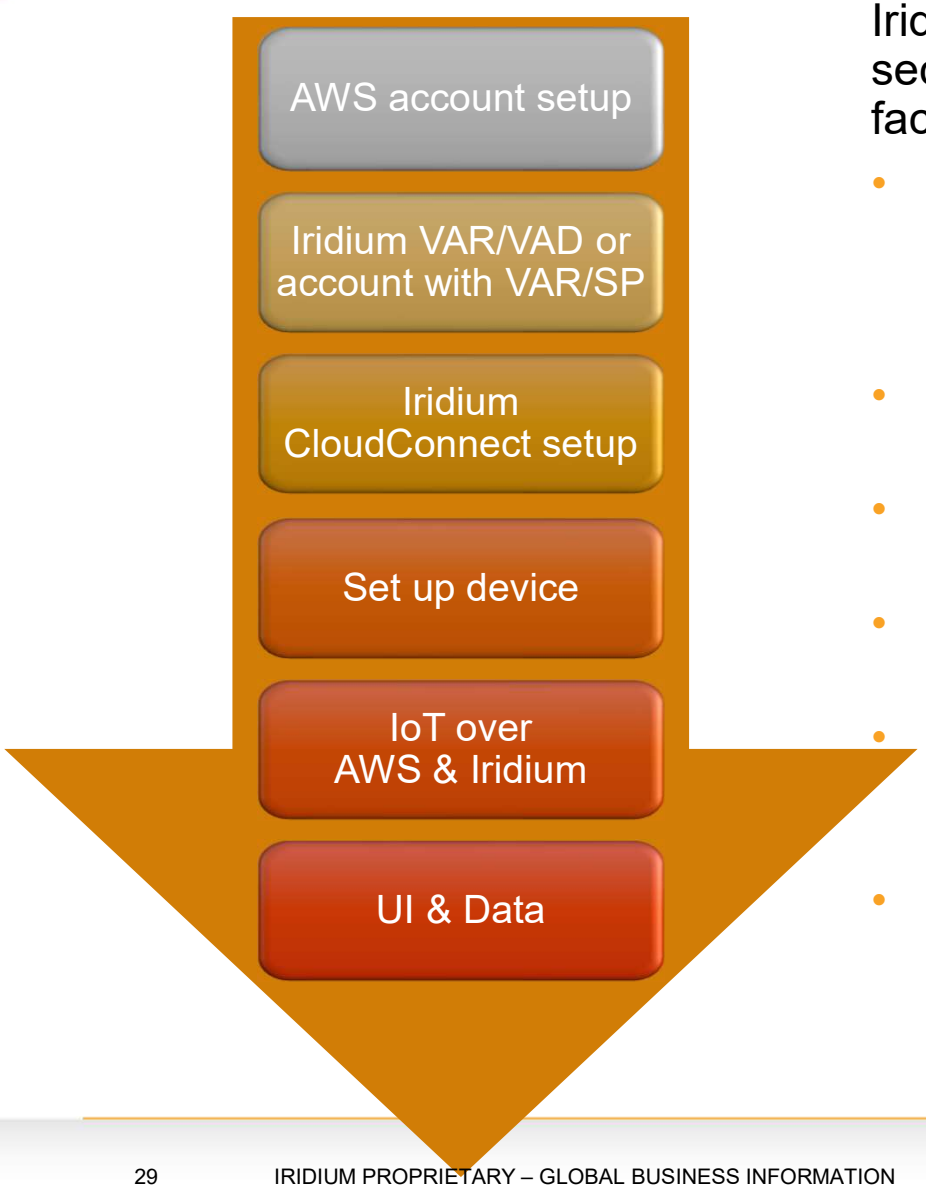
- Further integration with AWS Core IoT services
  - SQS to MQTT
  - Device management
- Offer SBD EMT/EMO with AWS based system as part of development kit offer
- AWS DirectConnect as alternative to VPN, Colo.
- RUDICS connections to AWS back ends
- Further IWS and Spnet integration

2020+: Future

- Support future Iridium messaging protocols
  - Certus Messaging
- Support additional cloud providers direct our through AWS solutions
  - Azure
  - Google
  - AliCloud
- Support additional IoT & industry protocols
  - AWS/GCP/AZI/ACI
- Unique Country and Customer restrictions
  - China
  - Classified

VAR CONSIDERATIONS

STEPS TO GET IRIDIUM "CLOUDCONNECTED" THROUGH A VAR



Iridium and customer set up cross-account security role using AWS console (IAM user), facilitated through VAR.

- Iridium provides CloudFormation template to VAR, who sends to customer. (creates all the AWS infrastructure in the Partner AWS EC2 instance)
- Customer provides URLs and cross-account ARN to Iridium (via VAR)
- Iridium completes setup and provide destination IPs
- VAR sets up SBD devices to use Iridium CloudConnect IP as destination (SPnet/IWS)
- Customer extracts/insert MO/MT from queues in their own AWS account instead of via DirectIP/Email/VPN
- Support calls fielded by VAR

CLOUDCONNECT VAR CONSIDERATIONS

- Initial briefing call by VAR
- Support calls fielded by VAR
- VAR sets up a new CloudConnect channel – Iridium does not record who it is for. For example VAR could have 20 ICC instances, and would need to know which customer is assigned to which instance. Iridium only records these are for VAR. (Similar to DirectIP today.)
- Troubleshooting and diagnostics by VAR
- Beyond initial POCs, Considering a CloudConnect certified partner moniker, based on AWS credentials.

FAQ: COSTS

- Customer is responsible for the cost of the AWS services used while running Iridium CloudConnect deployment.
- No additional transactional costs for providing SBD messages through CloudConnect (don’t charge for message twice.)
- Proposed one-time setup fee per CloudConnect instance to cover Iridium support burden. VAR may choose to pass this on to any end customer. TBC beyond POCs.