DYNAMODB LINK USER GUIDE

DSLINK-JAVA-V2-DynamoDB

CONTENTS

1. **Module** 2

1.1 **Connect to DynamoDB** 2

1.2 **Query Table** 3

1.3 **Scan** 7

1.4 **PutItem** 10

1.5 **Batch PutItem** 13

1.6 **UpdateItem** 14

1.7 **DeleteItem** 17

1. MODULE

Dynamo DB uses a NoSQL database model, which is no relational, allowing documents, graphs and columnar among its data models. Amazon DynamoDB is a fully managed NoSQL database service that allows to create database tables that can store and retrieve any amount of data.

1.1 CONNECT TO DYNAMODB

DynamoDB DS Link can connect with Dynamo DB using either one of the following options.

- AWS Cloud DynamoDB
- Local DynamoDB

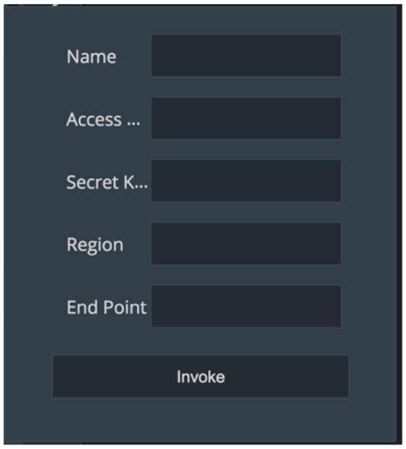
User has to provide the following details to connect DyanmoDB :

Field	Details
Name	• Any Name to identify dynamoDB
AWS Access Key	• AWS DynamoDB Access Key
Secret Access Key	• AWS DynamoDB Secret Key
Region	• DynamoDB Region
End Point	• Dynamod DB regional End Point

More information on Region and End Point HERE.

Example to connect to Local DynamoDB

- Name: Any Name
- AWS Access Key: You can give any string as its local DynamoDB
- Secrete Access key : Any string as its local DynamoDB
- Region: Select one of the valid Region based on installation. More details HERE.
- End Point : Http://localhost:8000



1.2 QUERY TABLE

The Query follows AWS's standard Query API. Following are details of each Query Fields. More details on Query attributes and values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.htm

- A single Query operation will read up to the **maximum number of items** set (if using the ***Limit*** parameter) or a maximum of **1 MB** of data and then apply any filtering to the results using ***FilterExpression***.
- If ***LastEvaluatedKey*** is present in the response, you will need to paginate the result set. For more information, see Paginating the Results in the Amazon DynamoDB Developer Guide.

Field	**Details**
Table Name	• **Required** • Table Name to query
Projection Expression	• **Optional** • Comma separated attributes names to retrieve from the table. • If no attribute names are specified, then all attributes will be returned.
Key Condition Expression	• **Required** • The condition that specifies the key value(s) for items to be retrieved by the Query action • The condition **must** perform an **equality** test on a **single partition key** value Example: partitionKeyName = :partitionkeyval • The condition can **optionally** perform one of several **comparison** tests on a **single sort key** value. For more comparison operator check HERE. Valid comparisons for the sort key condition are as follows:

	○ sortKeyName = :sortkeyval - true if the sort key value is equal to :sortkeyval. ○ sortKeyName < :sortkeyval - true if the sort key value is less than :sortkeyval. ○ sortKeyName <= :sortkeyval - true if the sort key value is less than or equal to:sortkeyval. ○ sortKeyName > :sortkeyval - true if the sort key value is greater than :sortkeyval. ○ sortKeyName >= :sortkeyval - true if the sort key value is greater than or equal to :sortkeyval. ○ sortKeyName BETWEEN :sortkeyval1 AND :sortkeyval2 - true if the sort key value is greater than or equal to :sortkeyval1, and less than or equal to :sortkeyval2. ○ begins_with (sortKeyName, :sortkeyval) - true if the sort key value begins with a particular operand. (You cannot use this function with a sort key that is of type Number.) Note that the function name begins_with is case-sensitive • Use **ExpressionAttributeValues** parameter to replace tokens such as :partitionval and :sortval with actual values at runtime. • Optionally use the **ExpressionAttributeNames** parameter to replace the names of the partition key and sort key with placeholder tokens Few Examples • partionKey= : partitionkeyval • partionKey= : partitionkeyval and sortKeyName < :sortkeyval
Filter Expression	• **Optional** • **Does not** allow key attributes - **partition key** or a **sort key**. This is applicable to **non** Partition and Sort key. • Conditions to be applied Query operation, but before the data is returned. • This is applied after the items have already been read; the process of filtering does not consume any additional read capacity units. • Few Examples: • AttributeName between :attributeval1 and :attributeval2 • AttributeName> :attributeval
Expression Attribute Names	• **Optional** (depends on **Key Condition Expression**) • Substitution tokens for attribute names in expression. • Used in case any expression uses DynamoDB **reserved** keywords. More details HERE. • Example {“#yr” : ”year” , "#P":"Percentile"}

	Use this substitution in an expression, as in this example: #yr = :val

Expression Attribute Values	- **Required** (depending on expression conditions)
	- Values that can be substituted in an expression. More details HERE.
	- Suppose that you wanted to check whether the value of the *ProductStatus* attribute was one of the following
	Available | Backordered | Discontinued
	Then use following in expression condition
	ProductStatus IN (:avail, :back, :disc)
	And then **ExpressionAttributeValues** as follows:
	{ ":avail":{"S":"Available"}, ":back":{"S":"Backordered"}, ":disc":{"S":"Discontinued"} }
	- Below is example for each data type. (B=Byte, BOOL=Boolean, BS=Byte Set, L=List, M=Map\Json, N=Number, NS=Number Set, S=String, SS=String Set)
	{
	":B": {"B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"},
	":BOOL": {"BOOL": true},
	":BS": {"BS":["U3Vubnk=", "UmFpbnk=", "U25vd3k="]},
	":L": {"L": ["Cookies", "Coffee", 3.14159]},
	":M": {"M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}},
	":N": {"N": "123.45"},
	":NS": {"NS": ["42.2", "-19", "7.5", "3.14"]},
	":NULL": {"NULL": true},
	":S": {"S": "Hello"},
	":SS": {"SS": ["Giraffe", "Hippo" ,"Zebra"]}
	}

Exclusive Start Key	- **Optional** (used for pagination) - The **primary key** of the first item that this operation will evaluate. - Use the value that was returned for **LastEvaluatedKey** in the previous operations result. - Example

	{"year":{"N":"2006"},"title":{"S":"All the King's Men"}} Note: 'year' is primary key and 'title' is sort key
Select	• **Optional** • The attributes to be returned in the result • Valid values are ○ ALL_ATTRIBUTES (default) ○ ALL_PROJECTED_ATTRIBUTES ○ COUNT ○ SPECIFIC_ATTRIBUTES • More details HERE. • For Example : • Select: ALL_ATTRIBUTES • Select: SPECIFIC_ATTRIBUTES
Limit	• **Optional** • Maximum Items to evaluate. • 0\null will return all items. • If there are more records, return result returns **Last Evaluated Key** so you pick up where it left. • Also if the processed size exceeds data size of 1MB **Last Evaluated Key** is returned. • For Example : • Limit: 5
ScanIndex Forward	• **Optional** • Default **true** • Specifies the order for index traversal
ConsistentRead,	• **Optional** • Default **false**. • Determines the read consistency model
ReturnConsumedCapacitys	• **Optional** • Determines the level of detail about provisioned throughput consumption that is returned in the response • Valid Values: INDEXES | TOTAL | NONE

Return Value is JSON with following fields

Field	Details
ITEMS	• Array of item attributes that match the query criteria • Empty Array if not Items found or **Select** attribute only '**COUNT**'.
Count	• The number of items in the response
ScannedCount	• The number of items evaluated, before any **QueryFilter** is applied

LastEvaluatedKey	- The **primary key** of the item where the operation stopped. - If the Query has more record to return but the result is limited because of Limit\filter and other attributes. - Typically used form pagination. - If LastEvaluatedKey is empty, then the "last page" of results has been processed and there is no more data to be retrieved. - If **LastEvaluatedKey** is not empty, it does not necessarily mean that there is more data in the result set. The only way to know when you have reached the end of the result set is when LastEvaluatedKey is empty. - Use this as input for **Exclusive Start Key** to query more items.

1.3 SCAN

The Scan follows AWS's standard Scan API. Following are details of each Scan Fields. More details on Scan attributes and values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Scan.html

- If the total number of scanned items exceeds the maximum data set size limit of **1 MB**, the scan stops and results are returned to the user as a **LastEvaluatedKey** value to continue the scan in a subsequent operation. The results also include the number of items exceeding the limit. A scan can result in no table data meeting the filter criteria.
- If ***LastEvaluatedKey*** is present in the response, you will need to paginate the result set. For more information, see Paginating the Results in the Amazon DynamoDB Developer Guide.
- Request accepts the following data in JSON format.

Field	**Details**
Table Name	- **Required** - Table Name to query
Projection Expression	- **Optional** - Comma separated attributes names to retrieve from the table. - If no attribute names are specified, then all attributes will be returned.
Filter Expression	- **Optional** - Conditions to be applied Query operation, but before the data is returned. - **Does not** allow key attributes - **partition key** or a **sort key**. - This is applied after the items have already been read; the process of filtering does not consume any additional read capacity units. - Few Examples: - AttributeName between :attributeval1 and :attributeval2 - AttributeName> :attributeval
Expression Attribute Names	- **Optional** (depends on **Key Condition Expression**)

	- Substitution tokens for attribute names in expression. - Used in case any expression use DynamoDB reserved keywords. More details HERE. - Example {“#yr” : ”year” , "#P":"Percentile"} Use this substitution in an expression, as in this example: - #yr = :val
ExpressionAttributeValues	- **Optional** (depending on expression conditions) - Values that can be substituted in an expression. More details HERE. - Suppose that you wanted to check whether the value of the *ProductStatus* attribute was one of the following Available | Backordered | Discontinued Then use following in expression condition ProductStatus IN (:avail, :back, :disc) And then **ExpressionAttributeValues** as follows: { ":avail":{"S":"Available"}, ":back":{"S":"Backordered"}, ":disc":{"S":"Discontinued"} } - Below is example for each data type. (B=Byte, BOOL=Boolean, BS=Byte Set, L=List, M=Map\Json, N=Number, NS=Number Set, S=String, SS=String Set) { ":B": {"B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"}, ":BOOL": {"BOOL": true}, ":BS": {"BS":["U3Vubnk=", "UmFpbnk=", "U25vd3k="]}, ":L": {"L": ["Cookies", "Coffee", 3.14159]}, ":M": {"M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}}, ":N": {"N": "123.45"}, ":NS": {"NS": ["42.2", "-19", "7.5", "3.14"]}, ":NULL": {"NULL": true}, ":S": {"S": "Hello"}, ":SS": {"SS": ["Giraffe", "Hippo" ,"Zebra"]}

	}
Exclusive Start Key	• **Optional** (used for pagination) • The **primary key** of the first item that this operation will evaluate. • Use the value that was returned for **LastEvaluatedKey** in the previous operations result. • Example {"year":{"N":"2006"},"title":{"S":"All the King's Men"}} Note: 'year' is primary key and 'title' is sort key
Select	• **Optional** • The attributes to be returned in the result • Valid values are ○ ALL_ATTRIBUTES (default) ○ ALL_PROJECTED_ATTRIBUTES ○ COUNT ○ SPECIFIC_ATTRIBUTES More details HERE. For Example : • Select: ALL_ATTRIBUTES • Select: SPECIFIC_ATTRIBUTES
Limit	• **Optional** • Maximum Items to evaluate. • 0 will return all items. • If there are more records, return result returns **Last Evaluated Key** so you pick up where it left. • Also if the processed size exceeds data size of 1MB **Last Evaluated Key** is returned. • For Example : Limit: 5
Segment	• **Optional** • The value for Segment must be **greater than or equal to 0**, and **less than** the value provided for **TotalSegments**. • The value of **LastEvaluatedKey** returned from a parallel Scan request must be used as **ExclusiveStartKey** with the same segment ID in a subsequent Scan operation. • If you provide Segment, you **must** also provide **TotalSegments**. Type : Integer. Valid Range : Min Value : 0 to Max Value : 999999
Total Segments	• **Optional**

	• If you specify TotalSegments, you **must also specify Segment**. • For a parallel Scan request, TotalSegments represents the total number of segments into which the Scan operation will be divided. The value of TotalSegments corresponds to the number of application workers that will perform the parallel scan. • Must be **greater than or equal to 1, and less than or equal to 1000000**. Type : Integer Valid Range : Min Value : 1 to Max Value 1000000
Consistent Read	• **Optional** • Default false. • Determines the read consistency model
Return Consumed Capacity	• **Optional** • Determines the level of detail about provisioned throughput consumption that is returned in the response • Valid Values: INDEXES | TOTAL | NONE

Return Value is JSON with following fields

Field	**Details**
ITEMS	• Array of item attributes that match the query criteria • Empty Array if not Items found or **Select** attribute only '**COUNT**'.
Count	• The number of items in the response
ScannedCount	• The number of items evaluated, before any **QueryFilter** is applied
LastEvaluatedKey	• The **primary key** of the item where the operation stopped. - If the Query has more record to return but the result is limited because of Limit\filter and other attributes. • Typically used form pagination. • If LastEvaluatedKey is empty, then the "last page" of results has been processed and there is no more data to be retrieved. • If **LastEvaluatedKey** is not empty, it does not necessarily mean that there is more data in the result set. The only way to know when you have reached the end of the result set is when LastEvaluatedKey is empty. • Use this as input for **Exclusive Start Key** to query more items.

1.4 PUTITEM

The PutItem follows AWS's standard PutItem API. Following are details of each PutItem Fields. More details on each attribute and it's values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_PutItem.html

- If an item that has the **same primary key** as the new item already exists in the specified table, the new item completely **replaces** the existing item
- When you add an item, the primary key attribute(s) are the only required attributes.
- Attribute values **cannot be null**. String and Binary type attributes must have lengths greater than zero.
- Set type attributes (like String Set etc)cannot be empty. Requests with empty values will be rejected with a **ValidationException** exception.

Field	Details
Table Name	• Required

Item	• Required
	• Each element in the Item map is an **AttributeValue** object.Map of attribute name/value pairs, one for each attribute.
	• You **must** provide **all** of the attributes for the **primary key**.- For simple primary key provide value for only **partition key**. For composite provide **partition key and sort key**.
	• If you specify any attributes that are part of an index key, then the data types for those attributes must **match** those of the **schema in the table**'s attribute definition.
	Example
	{
	“id”: 23,
	“metrics”: 34.2 ,
	“sensorname”: “sensor1”
	}

Condition Expression	• **Optional** • A condition that must be satisfied in order for a conditional PutItem operation to succeed. • An expression can contain any of the following: ○ Functions: attribute_exists | attribute_not_exists | attribute_type | contains | begins_with | size ○ These function names are case-sensitive. ○ Comparison operators: = | <> | < | > | <= | >= | BETWEEN | IN ○ Logical operators: AND | OR | NOT For Example: condition expression : attribute_not_exists(AttributeName) condition expression : AttributeName between :attributeval1 and :attributeval2
Expression Attribute Names	• **Optional** (depends on **Condition Expression**) • Substitution tokens for attribute names in expression.

	• Used in case any expression use DynamoDB reserved keywords. More details HERE. • Example {“#yr” : ”year” , "#P":"Percentile"} Use this substitution in an expression, as in this example: #yr = :val

Expression Attribute Values	• **Optional** (depending on expression conditions)
	• Values that can be substituted in an expression. More details HERE.
	• Suppose that you wanted to check whether the value of the *ProductStatus* attribute was one of the following
	Available | Backordered | Discontinued
	Then use following in expression condition
	ProductStatus IN (:avail, :back, :disc)
	And then **ExpressionAttributeValues** as follows:
	{ ":avail":{"S":"Available"}, ":back":{"S":"Backordered"}, ":disc":{"S":"Discontinued"} }
	• Below is example for each data type. (B=Byte, BOOL=Boolean, BS=Byte Set, L=List, M=Map\Json, N=Number, NS=Number Set, S=String, SS=String Set)
	{
	":B": {"B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"},
	":BOOL": {"BOOL": true},
	":BS": {"BS":["U3Vubnk=", "UmFpbnk=", "U25vd3k="]},
	":L": {"L": ["Cookies", "Coffee", 3.14159]},
	":M": {"M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}},
	":N": {"N": "123.45"},
	":NS": {"NS": ["42.2", "-19", "7.5", "3.14"]},
	":NULL": {"NULL": true},
	":S": {"S": "Hello"},
	":SS": {"SS": ["Giraffe", "Hippo" ,"Zebra"]}
	}

Return Value:

Field	Details
Result	• If successful returns Item inserted

1.5 BATCH PUTITEM

The Batch PutItem follows AWS's standard BatchWriteItem API. Following are details of each fields. More details on each attribute and it's values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_BatchWriteItem.html

- A single call to Batch PutItem can write up to **16 MB** of data, which can comprise as many as **25 put** requests. Individual items to be written can be as large as **400 KB**.
- The Batch PutItem takes care of Unproccessed Items internally.
- If NONE of the items can be processed due to insufficient provisioned throughput on all of the tables in the request, then **BatchWriteItem** will return a **Provisioned Throughput Exceeded Exception**.

If one or more of the following is true, DynamoDB rejects the entire batch write operation:

- Primary key attributes specified on an item in the request do not match those in the corresponding table's primary key schema.
- Input Items list contains at least two items with identical hash and range keys (which essentially is two put operations).
- There are more than 25 requests in the batch.
- Any individual item in a batch exceeds 400 KB.
- The total request size exceeds 16 MB.

Field	Details
Table Name	• Required
Items	• Required

	• All the Attributevalue objects should be given in an array like : [{Item 1…..}, {Item2…..},{Item3……..},{Item4…..}] [{“id”: 23, “metrics”: 34.2 , “sensorname”: “sensor1”}, {“id”: 34, “metrics”: 40.2 , “sensorname”: “sensor3”}, {“id”: 32, “metrics”: 50.2 , “sensorname”: “sensor4”}] • Each element in the Item map is an **AttributeValue** object.Map of attribute name/value pairs, one for each attribute. • You **must** provide **all** of the attributes for the **primary key**.- For simple primary key provide value for only **partition key**. For composite provide **partition key and sort key**. • If you specify any attributes that are part of an index key, then the data types for those attributes must **match** those of the **schema in the table**'s attribute definition.

Return Value

Field	Details
Result	• If successful returns Array of Items • If failed nothing returned

1.6 UPDATEITEM

The updateItem follows AWS’s standard UpdateItem API. Following are details of each UpdateItem Fields. More details on each attribute and it’s values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_UpdateItem.html

- Updates only **one** Item
- **Edits** an existing item's attributes, **or adds a new item** to the table if it does not already exist.
- You can put, delete, or add attribute values.
- You can also perform a conditional update on an existing item (insert a new attribute name-value pair if it doesn't exist, or replace an existing name-value pair if it has certain expected attribute values).

Field	Details
Table Name	• **Required**

	• Table Name of Item to be updated.

Primary Key	• **Required**
	• The primary key of the item to be updated. For a **composite primary key**, you must provide values for both the **partition key and the sort key**.
	• Example (Composite Primary Key):
	{
	"primarykey": {"S": "primarykeyvalue"},
	"sortkey": {"S": "sort key value"}
	}

Update Expression	• **Optional** • SET - adds one or more attributes and values to an item. If any of these attribute already exist, they are replaced by the new values. • For Example: SET AttributeName= :attributeval • REMOVE - Removes one or more attributes from an item. • For Example: Remove AttributeName= :attributeval • ADD - Adds the specified value to the item, if the attribute does not already exist. If it exists the result is based on type (Only supports **Number** of **Set** of same type) ○ If type if **Number**. It adds to previous value ○ If it I of type **Set** then value is added into the Set. • For Example: ADD AttributeName= :attributeval • DELETE - Deletes an element from a set. For Example: DELETE AttributeName= :attributeval
Condition Expression	• **Optional** • A condition that must be satisfied in order for a conditional PutItem operation to succeed. • An expression can contain any of the following: ○ Functions: attribute_exists | attribute_not_exists | attribute_type | contains | begins_with | size ○ These function names are case-sensitive. ○ Comparison operators: = | <> | < | > | <= | >= | BETWEEN | IN ○ Logical operators: AND | OR | NOT For Example: • condition expression : attribute_not_exists(AttributeName) • condition expression : AttributeName between :attributeval1 and :attributeval2
Expression Attribute Names	• **Optional** (depends on **Condition Expression**)

	• Substitution tokens for attribute names in expression. • Used in case any expression use DynamoDB reserved keywords. More details HERE. • Example {“#yr” : ”year” , "#P":"Percentile"} Use this substitution in an expression, as in this example: #yr = :val

ExpressionAttributeValues	• **Optional** (depending on expression conditions)
	• Values that can be substituted in an expression. More details HERE.
	• Suppose that you wanted to check whether the value of the *ProductStatus* attribute was one of the following
	Available | Backordered | Discontinued
	Then use following in expression condition
	ProductStatus IN (:avail, :back, :disc)
	And then **ExpressionAttributeValues** as follows:
	{ ":avail":{"S":"Available"}, ":back":{"S":"Backordered"}, ":disc":{"S":"Discontinued"} }
	• Below is example for each data type.
	{
	":B": {"B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"},
	":BOOL": {"BOOL": true},
	":BS": {"BS":["U3Vubnk=", "UmFpbnk=", "U25vd3k="]},
	":L": {"L": ["Cookies", "Coffee", 3.14159]},
	":M": {"M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}},
	":N": {"N": "123.45"},
	":NS": {"NS": ["42.2", "-19", "7.5", "3.14"]},
	":NULL": {"NULL": true},
	":S": {"S": "Hello"},
	":SS": {"SS": ["Giraffe", "Hippo" ,"Zebra"]}
	}

Return Value:

Field	Details
Result	- If successful returns updated Items - If failed returns nothing

1.7 DELETEITEM

The DeleteItem follows AWS's standard DeleteItem API. Following are details of each DeleteItem Fields. More details on each attribute and it's values - https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_DeleteItem.html

- Deletes a **single** item in a table by primary key.
- Unless you specify conditions, the **DeleteItem** is an idempotent operation; running it multiple times on the same item or attribute **does NOT result in an error response**.
- Conditional deletes are useful for deleting items only if specific conditions are met. If those conditions are met, DynamoDB performs the delete. Otherwise, the item is not deleted.

Field	Details
Table Name	- **Required** The name of the table from which to delete the item.
Primary Key	- **Required** - The primary key of the item to be updated. Each element consists of an attribute name and a value for that attribute. - For a composite primary key, you must provide values for both the partition key and the sort key.
Condition Expression	- **Optional** - A condition that must be satisfied in order for a conditional PutItem operation to succeed. - An expression can contain any of the following:

	1. Functions: attribute_exists | attribute_not_exists | attribute_type | contains | begins_with | size These function names are case-sensitive. 2. Comparison operators: = | <> | < | > | <= | >= | BETWEEN | IN 3. Logical operators: AND | OR | NOT For Example: condition expression : attribute_not_exists(AttributeName) condition expression : AttributeName between :attributeval1 and :attributeval2
Expression AttributeNames	• **Optional** (depends on **Condition Expression**) • Substitution tokens for attribute names in expression. • Used in case any expression use DynamoDB reserved keywords. More details HERE. • Example {“#yr” : ”year” , "#P":"Percentile"} Use this substitution in an expression, as in this example: #yr = :val
ExpressionAttributeValues	• **Optional**(depending on expression conditions) • Values that can be substituted in an expression. More details HERE. • Suppose that you wanted to check whether the value of the *ProductStatus* attribute was one of the following Available | Backordered | Discontinued Then use following in expression condition ProductStatus IN (:avail, :back, :disc) And then **ExpressionAttributeValues** as follows: { ":avail":{"S":"Available"}, ":back":{"S":"Backordered"}, ":disc":{"S":"Discontinued"} } • Below is example for each data type. {

	":B": {"B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"}, ":BOOL": {"BOOL": true}, ":BS": {"BS":["U3Vubnk=", "UmFpbnk=", "U25vd3k="]}, ":L": {"L": ["Cookies", "Coffee", 3.14159]}, ":M": {"M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}}, ":N": {"N": "123.45"}, ":NS": {"NS": ["42.2", "-19", "7.5", "3.14"]}, ":NULL": {"NULL": true}, ":S": {"S": "Hello"}, ":SS": {"SS": ["Giraffe", "Hippo" ,"Zebra"]} }

Return Value:

Field	**Details**
Result	• JSON Array of Items