DynamoDB support for .NET source code

CAST supports **DynamoDB** via its NoSQL for .NET extension. Details about how this support is provided for **.NET source code** is discussed below.

Supported Client Libraries

AWS DynamoDB .NET SDK	

Supported Operations

Operation	Methods Supported
Insert	• Amazon.DynamoDBv2.DocumentModel.Table.PutItem • Amazon.DynamoDBv2.DocumentModel.Table.PutItemAsync • Amazon.DynamoDBv2.DocumentModel.Table.TryPutItem • Amazon.DynamoDBv2.DataModel.DynamoDBContext.Save • Amazon.DynamoDBv2.DataModel.DynamoDBContext.SaveAsync
Update	• Amazon.DynamoDBv2.AmazonDynamoDBClient.UpdateItem • Amazon.DynamoDBv2.AmazonDynamoDBClient.UpdateItemAsync • Amazon.DynamoDBv2.AmazonDynamoDBClient.UpdateTable • Amazon.DynamoDBv2.AmazonDynamoDBClient.UpdateTableAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.UpdateItem • Amazon.DynamoDBv2.IAmazonDynamoDB.UpdateItemAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.UpdateTable • Amazon.DynamoDBv2.IAmazonDynamoDB.UpdateTableAsync • Amazon.DynamoDBv2.DocumentModel.Table.UpdateItem • Amazon.DynamoDBv2.DocumentModel.Table.UpdateItemAsync • Amazon.DynamoDBv2.DocumentModel.Table.TryUpdateItem

Select	• Amazon.DynamoDBv2.AmazonDynamoDBClient.Scan • Amazon.DynamoDBv2.AmazonDynamoDBClient.ScanAsync • Amazon.DynamoDBv2.AmazonDynamoDBClient.BatchGetItem • Amazon.DynamoDBv2.AmazonDynamoDBClient.BatchGetItemAsync • Amazon.DynamoDBv2.AmazonDynamoDBClient.BatchWriteItem • Amazon.DynamoDBv2.AmazonDynamoDBClient.BatchWriteItemAsync • Amazon.DynamoDBv2.AmazonDynamoDBClient.GetItem • Amazon.DynamoDBv2.AmazonDynamoDBClient.GetItemAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.DescribeTable • Amazon.DynamoDBv2.IAmazonDynamoDB.DescribeTableAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.BatchGetItem • Amazon.DynamoDBv2.IAmazonDynamoDB.BatchGetItemAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.Query • Amazon.DynamoDBv2.IAmazonDynamoDB.QueryAsync • Amazon.DynamoDBv2.DataModel.DynamoDBContext.QueryAsync • Amazon.DynamoDBv2.DataModel.DynamoDBContext.LoadAsync • Amazon.DynamoDBv2.DataModel.DynamoDBContext.Load • Amazon.DynamoDBv2.DataModel.DynamoDBContext.Scan • Amazon.DynamoDBv2.DataModel.DynamoDBContext.ScanAsync • Amazon.DynamoDBv2.DataModel.DynamoDBContext.FromDocuments • Amazon.DynamoDBv2.DataModel.DynamoDBContext.FromQuery • Amazon.DynamoDBv2.DocumentModel.Table.DescribeTable • Amazon.DynamoDBv2.DocumentModel.Table.GetItem • Amazon.DynamoDBv2.DocumentModel.Table.GetItemAsync • Amazon.DynamoDBv2.DocumentModel.Table.Query • Amazon.DynamoDBv2.DocumentModel.Table.Scan • Amazon.DynamoDBv2.DocumentModel.Table.TryLoadTable
Delete	• Amazon.DynamoDBv2.AmazonDynamoDBClient.DeleteItem • Amazon.DynamoDBv2.AmazonDynamoDBClient.DeleteItemAsync • Amazon.DynamoDBv2.AmazonDynamoDBClient.DeleteTable • Amazon.DynamoDBv2.AmazonDynamoDBClient.DeleteTableAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.DeleteItem • Amazon.DynamoDBv2.IAmazonDynamoDB.DeleteItemAsync • Amazon.DynamoDBv2.IAmazonDynamoDB.DeleteTable • Amazon.DynamoDBv2.IAmazonDynamoDB.DeleteTableAsync • Amazon.DynamoDBv2.DataModel.DynamoDBContext.Delete • Amazon.DynamoDBv2.DataModel.DynamoDBContext.DeleteAsync • Amazon.DynamoDBv2.DocumentModel.Table.DeleteItem • Amazon.DynamoDBv2.DocumentModel.Table.DeleteItemAsync • Amazon.DynamoDBv2.DocumentModel.Table.TryDeleteItem

Objects

Icon	Description
	DotNet AWS DynamoDB Client
	DotNet AWS DynamoDB Table
	DotNet AWS Unknown DynamoDB Client
	DotNet AWS Unknown DynamoDB Table

Links

Links are created for transaction and function point needs:

Link type	Source and destination of link	Methods Supported

belongsTo	From DotNet AWS DynamoDB Table to DotNet AWS DynamoDB Client object	
useInsertink	Between the caller .NET Method (C#) object and DotNet AWS DynamoDB Table	• PutItem • PutItemAsync • TryPutItem • Save • SaveAsync
useUpdateLink		• UpdateTable • UpdateTableAsync • UpdateItem • UpdateItemAsync • TryUpdateItem
useSelectLink		• Query • QueryAsync • Scan • ScanAsync • DescribeTable • GetItem • GetItemAsync • BatchGetItem • BatchGetItemAsync • BatchWriteItem • BatchWriteItemAsync • Load • LoadAsync • FromDocuments • FromQuery • TryLoadTable
useDeleteLink		• DeleteItem • DeleteItemAsync • DeleteTable • DeleteTableAsync • Delete • TryDeleteItem

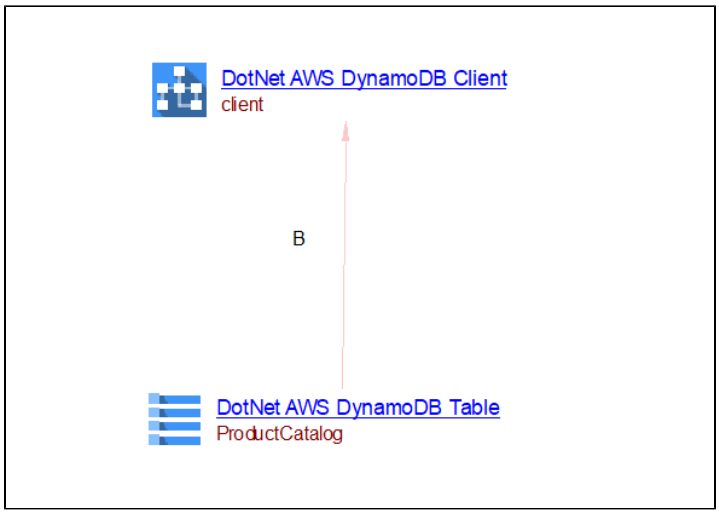
What results can you expect?

Once the analysis/snapshot generation has completed, you can view the results in the normal manner (for example via CAST Enlighten). Some examples are shown below.

AmazonDynamoDB Client and Table Creation

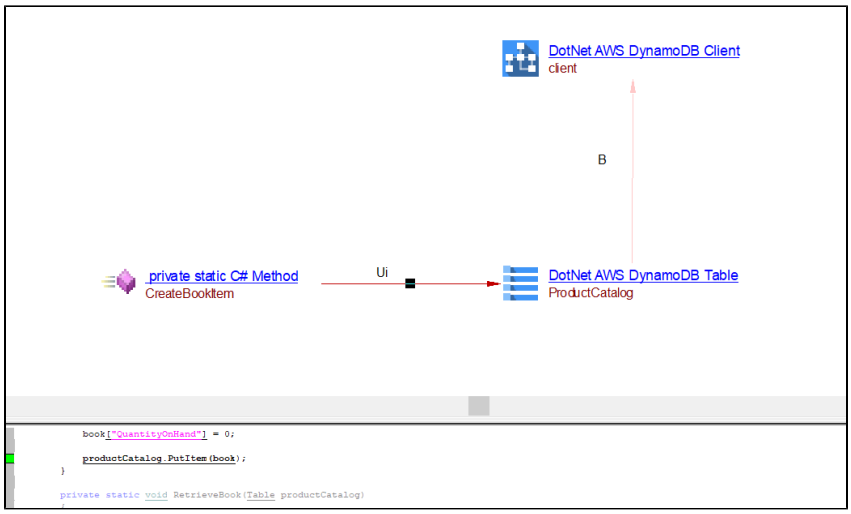
Client and Table creation

```
class MidlevelItemCRUD
    {
        AmazonDynamoDBConfig clientConfig = new AmazonDynamoDBConfig();
                // Set the endpoint URL
                clientConfig.ServiceURL = "http://localhost:8000";
                AmazonDynamoDBClient client = new AmazonDynamoDBClient(clientConfig);
        private static string tableName = "ProductCatalog";
        // The sample uses the following id PK value to add book item.
        private static int sampleBookId = 555;

        static void Main(string[] args)
        {
            try
            {
                Table productCatalog = Table.LoadTable(client, tableName);
                CreateBookItem(productCatalog);
                RetrieveBook(productCatalog);
                // Couple of sample updates.
                UpdateMultipleAttributes(productCatalog);
                UpdateBookPriceConditionally(productCatalog);

                // Delete.
                DeleteBook(productCatalog);
                Console.WriteLine("To continue, press Enter");
                Console.ReadLine();
            }
            catch (AmazonDynamoDBException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }
```



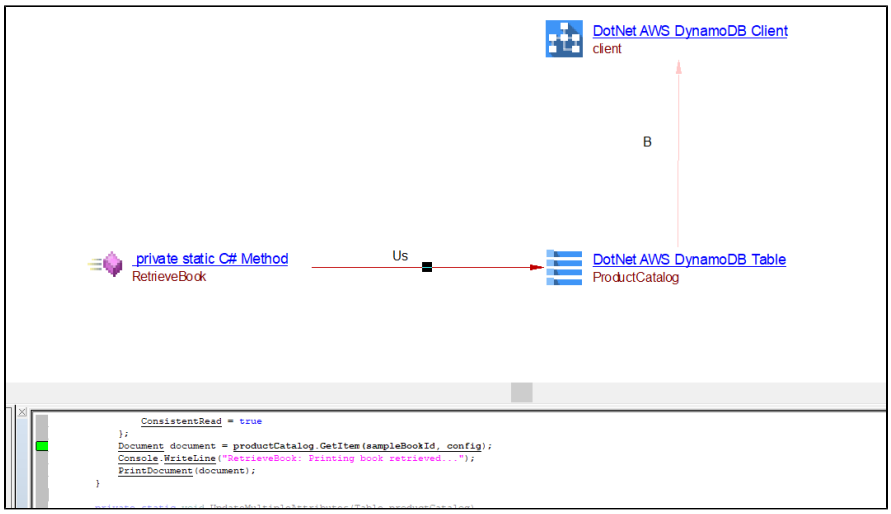
Insert Operation

```
private static void CreateBookItem(Table productCatalog)
        {
            Console.WriteLine("\n*** Executing CreateBookItem() ***");
            var book = new Document();
            book["Id"] = sampleBookId;
            book["Title"] = "Book " + sampleBookId;
            book["Price"] = 19.99;
            book["ISBN"] = "111-1111111111";
            book["Authors"] = new List<string> { "Author 1", "Author 2", "Author 3" };
            book["PageCount"] = 500;
            book["Dimensions"] = "8.5x11x.5";
            book["InPublication"] = new DynamoDBBool(true);
            book["InStock"] = new DynamoDBBool(false);
            book["QuantityOnHand"] = 0;

            productCatalog.PutItem(book);
        }
```



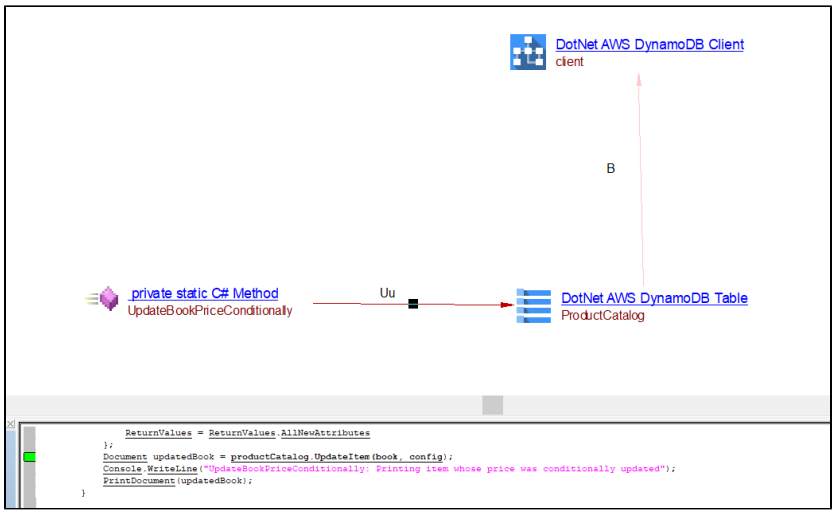
Select Operation

```
private static void RetrieveBook(Table productCatalog)
        {
            Console.WriteLine("\n*** Executing RetrieveBook() ***");
            // Optional configuration.
            GetItemOperationConfig config = new GetItemOperationConfig
            {
                AttributesToGet = new List<string> { "Id", "ISBN", "Title", "Authors", "Price" },
                ConsistentRead = true
            };
            Document document = productCatalog.GetItem(sampleBookId, config);
            Console.WriteLine("RetrieveBook: Printing book retrieved...");
            PrintDocument(document);
        }
```



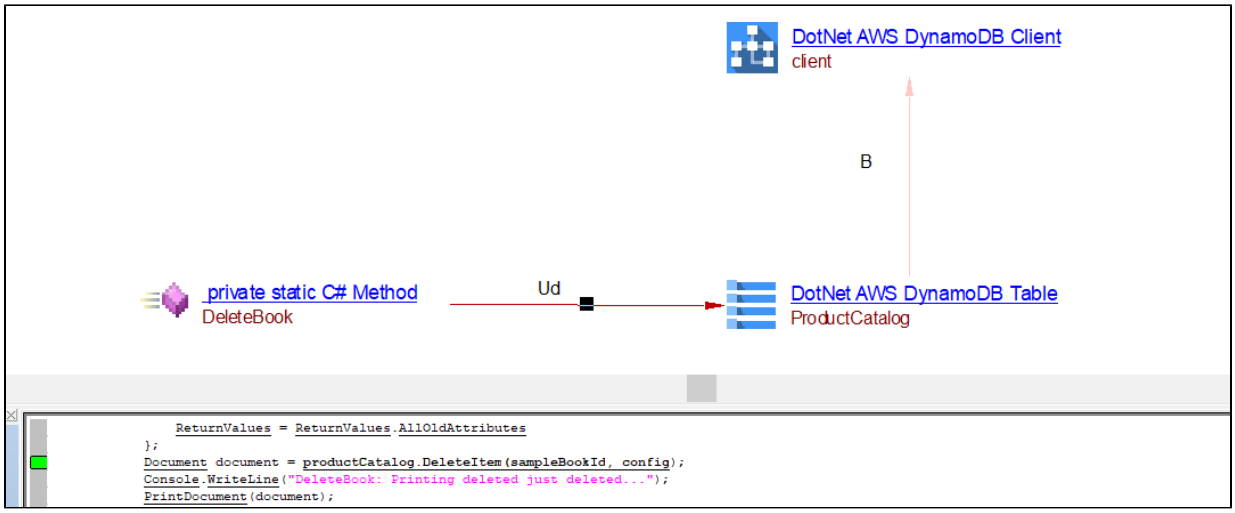
Update Operation

```
private static void UpdateBookPriceConditionally(Table productCatalog)
        {
            Console.WriteLine("\n*** Executing UpdateBookPriceConditionally() ***");

            int partitionKey = sampleBookId;

            var book = new Document();
            book["Id"] = partitionKey;
            book["Price"] = 29.99;

            // For conditional price update, creating a condition expression.
            Expression expr = new Expression();
            expr.ExpressionStatement = "Price = :val";
            expr.ExpressionAttributeValues[":val"] = 19.00;

            // Optional parameters.
            UpdateItemOperationConfig config = new UpdateItemOperationConfig
            {
                ConditionalExpression = expr,
                ReturnValues = ReturnValues.AllNewAttributes
            };
            Document updatedBook = productCatalog.UpdateItem(book, config);
            Console.WriteLine("UpdateBookPriceConditionally: Printing item whose price was conditionally
updated");
            PrintDocument(updatedBook);
        }
```



Delete Operation

```
private static void DeleteBook(Table productCatalog)
        {
            Console.WriteLine("\n*** Executing DeleteBook() ***");
            // Optional configuration.
            DeleteItemOperationConfig config = new DeleteItemOperationConfig
            {
                // Return the deleted item.
                ReturnValues = ReturnValues.AllOldAttributes
            };
            Document document = productCatalog.DeleteItem(sampleBookId, config);
            Console.WriteLine("DeleteBook: Printing deleted just deleted...");
            PrintDocument(document);
        }
```



Evolution

- Resolution for table objects and crud links are increased
- New support for Communication Models 'Amazon.DynamoDBv2.DocumentModel' and 'Amazon.DynamoDBv2.DataModel'

Limitations

- Client objects are always default with the name 'client'
- If the client and table are not resolved will create Unknown Objects.