	Incidencia – Reinstalaciòn de cajas (caja longeva no termina de instalar)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	31-10-2024	31-10-2024

Control de versiones

Versión	**Fecha**	**Autor**	**Descripción**
1.0	31-10-2024	Eduardo Cuevas	Se propone optimizar el proceso de extracción de registros desde la base de datos Central para cada tabla, ajustando la fecha de inicio en las consultas de extracción de datos. Esto permitirá reducir el volumen de datos procesados en cada operación, mejorando la eficiencia y velocidad del proceso.

GET Technologies	Incidencia – Reinstalaciòn de cajas (caja longeva no termina de instalar)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	31-10-2024	31-10-2024

Descripción general

Actualmente, en el servicio del Store, existe un grupo de tablas de tipo *upload* que envían datos de la sucursal a la base de datos **Central**. Durante el proceso de reinstalación o restauración de estas tablas del grupo ***upload***, al consultar la base de datos Central para extraer los datos que se restaurarán en la base de datos **Store**, se utiliza un parámetro de fecha fijado en ***1970-01-01***. Esto implica que se recuperen todos los registros de cada tabla en el grupo ***upload***, lo cual extiende los tiempos de **restauración** y puede ocasionar bloqueos en la base de datos o en las tablas afectadas.

Diagnóstico

El método **DownloadSendDataOneTime** es responsable de restaurar los datos del grupo de Upload. A medida que se ejecuta, construye dinámicamente el **request** de cada tabla que integra el grupo, permitiendo así consultar en la base de datos central los registros necesarios para la restauración.

```
SYNC_CATALOG_REQUEST_V3 request = new SYNC_CATALOG_REQUEST_V3();
request.MappingName = syncMapping.MappingName;
request.CurrentCount = 0;
request.CurrentLastUpdateDateTime = SyncServiceConfig.DefaultInitDateTime;
request.StoreId = a_storeId;
```

Como se muestra en la imagen anterior, en la propiedad **CurrentLastUpdateDateTime** del **request** se asigna el valor de la constante **SyncServiceConfig.DefaultInitDateTime**, cuyo valor es **1 de enero de 1970.**

```
public static DateTime DefaultInitDateTime = new DateTime(1970, 1, 1, 0, 0, 0, 0);
```

Esto provoca dos problemas durante la ejecución de la petición:

1. **Consulta de toda la información**: Al establecer esta fecha por defecto, se trae toda la información almacenada de la sucursal en la base de datos **Central** para la tabla en proceso, lo cual genera una sobrecarga de datos innecesaria.
2. **Sobrecarga de memoria**: La gran cantidad de datos extraídos se almacena temporalmente en memoria, lo que en repetidas ocasiones causa una excepción de

tipo OutOfMemoryException. Como resultado, el servicio se detiene y necesita ser reiniciado en múltiples ocasiones para continuar su operación.

Debido a los problemas mencionados, el servicio en Store no logra completar sus tareas satisfactoriamente. Cada vez que falla en restaurar los datos, el servicio intenta nuevamente realizar la tarea de restauración, lo que resulta en un proceso lento y, en muchos casos, en una falla continua al arrojar el mismo error de forma repetida.

Posible solución

Actualmente, la tabla **CNFPARAMETROSCONFIGURACIONAPLICACION** contiene registros que permiten la depuración de datos en rangos de tiempo específicos. Por ejemplo, el Id 1097 (valor 8) y el Id 10970 (2/M|OPCOLAOPERACIONES;) Se usan para el caso de Store. Estos registros se replican en la tabla AX_parameters, donde el Id 510 referencia al Id 1097, mientras que los Ids 520 y 521 refieren al Id 10970.

El registro en **AX_parameters** con Id 510 se usa como valor predeterminado para depurar tablas, con un rango de 8 meses (fecha actual - 8 meses). Por ejemplo, si la fecha es 19-01-24, se eliminan registros anteriores a 19-05-23. Para los Ids 520 y 521, si existen en la base de datos Store, aplican un rango especial. El Id 520, por ejemplo, tiene el formato 3/M|OPCOLAOPERACIONES,GENBITACORAMONEDERO,RTPPOSFDFINALOR;, donde 3/M significa 3 meses para las tablas listadas. El Id 521 usa W para semanas, como 6/W|cnfOperacionesCat,venTurnosCajaReg; (6 semanas).

Con esta configuración, los períodos de purga predeterminados (Id 510) y especiales (520, 521) se emplean como fecha de inicio para consultar datos en la base de datos Central y restaurar las tablas en **Store**. Esto optimiza el proceso al evitar consultar desde 1970, lo cual reducía considerablemente el tiempo de **restauración**

Además, este enfoque ofrece otro beneficio importante: al reducir el período de tiempo, se obtienen menos registros desde la base de datos Central hacia Store, lo que significa menos datos en memoria para insertar en Store. Esto contribuye a una mayor eficiencia en el proceso y reduce el riesgo de sobrecarga de memoria.

A continuación se muestran las imágenes de los métodos implementados para extraer la **fecha de inicio**, adaptándose según la tabla en proceso de **restauración**.

	Incidencia – Reinstalaciòn de cajas (caja longeva no termina de instalar)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	31-10-2024	31-10-2024

```
//Determina el periodo de fecha de inicio para la extraciòn de datos en base si establa especial o no.
var IsTableSpecial = false;
DateTime valuePeriod = SyncStoreTools.GetInitDate(a_tools, listAxParam, syncMapping.MappingObject.TableName.ToUpper(),
                                                  out a_defaultParameterRange, out IsTableSpecial);

Logger.AddInfoSubModule(SyncStoreService.SubModule,
    AxNLS.Data(1153000263, 2110, "[SYNC_INIT_DATA] Se descargara la tabla:'{0}' con el servidor central, " +
                                 "a partir de la fecha:{1}, IsTableSpecial:{2}", syncMapping.MappingObject.TableName, valuePeriod.ToString("yyyy-MM-dd"), IsTableSpecial));
```

Se crea el método **GetInitDate** en la clase **SyncStoreTools**, el cual tiene la función de consultar los registros de la tabla AX_parameters en función de la tabla en proceso. Si la tabla actual pertenece a la lista de tablas especiales (Ids 520 o 521), se obtiene la fecha correspondiente; de lo contrario, se utiliza el valor predeterminado del Id 510.

```
/// <returns></returns>
1 referencia
public static DateTime GetInitDate(DatabaseTools a_tools, IList<AX_PARAMETERS> parameters, string NameTable, out int a_defaultParameterRange, out bool IstableSpecial)
{
    a_defaultParameterRange = 0;
    IstableSpecial = false;

    var dateCurrent = DateTime.Now;
    string defaultValue = parameters.Where(x => x.PARAMETER == 510)
                                    .Select(x => x.VALUE)
                                    .FirstOrDefault();

    //parametro para en caso de ser tabla especial, este se inserte x veces por bloques a store
    a_defaultParameterRange = Convert.ToInt32(defaultValue);

    string valuePurgeTables = parameters.Where(x => x.PARAMETER != 510
                                        && x.VALUE.ToUpper().Contains(NameTable))
                                    .Select(x => x.VALUE)
                                    .FirstOrDefault() ?? "";

    //si no existe registros en AX_Parameters 520/521 de purgado especial, toma el default de 510
    if (string.IsNullOrEmpty(valuePurgeTables))
    {
        return dateCurrent.AddMonths(-(Convert.ToInt32(defaultValue)));
    }
    else
    {
        string patron = @"(\d+)/(M|W)";
        Regex regex = new Regex(patron);
        Match match = regex.Match(valuePurgeTables);

        string valueParameter = match.Groups[1].Value;
        string typeParameter = match.Groups[2].Value;

        IstableSpecial = true;

        if (typeParameter == "M")//MES
        {
            return dateCurrent.AddMonths(-(Convert.ToInt32(valueParameter)));
        }
        else//SEMANA
        {
            var days = 7 * (Convert.ToInt32(valueParameter));
            return dateCurrent.AddDays(-days);
        }
    }
}
```

De esta manera, el método determina la fecha de inicio necesaria para realizar la consulta para la tabla en curso.

```
//Determina el periodo de fecha de inicio para la extracìon de datos en base si establa especial o no.
var IsTableSpecial = false;
DateTime valuePeriod = SyncStoreTools.GetInitDate(a_tools, listAxParam, syncMapping.MappingObject.TableName.ToUpper(),
                                                  out a_defaultParameterRange, out IsTableSpecial);


Logger.AddInfoSubModule(SyncStoreService.SubModule,
    AxNLS.Data(1153000263, 2110, "[SYNC_INIT_DATA] Se descargara la tabla:'{0}' con el servidor central, " +
                                 "a partir de la fecha:{1}, IsTableSpecial:{2}", syncMapping.MappingObject.TableName, valuePeriod.ToString("yyyy-MM-dd"), IsTableSpecial));


SYNC_CATALOG_REQUEST_V3 request = new SYNC_CATALOG_REQUEST_V3();
request.MappingName = syncMapping.MappingName;
request.CurrentCount = 0;
request.CurrentLastUpdateDateTime = valuePeriod;
request.StoreId = a_storeId;

SyncCatalogClient client = new SyncCatalogClient(a_client);
if ((ns = client.SyncCatalog(request, SyncCatalogClient.SyncTypes.Init, out ns)).IsFaulted)
```