**Scheduled Packages:**

* **Sales Data** (D:\Inout\In\Sales Data\SalesData-Schedulings)
* **Direct upload of companies data from Cokpit DB** (D:\Inout\In\Daily-Upload-Companies-From-Cokpit DB\Daily-Upload-Companies-From-Cokpit DB\Package.dtsx)
* **Update Cr Manager in Cokpit DB** (D:\Inout\In\Cokpit-WareHouse-CRM\Cr Manager Updation\CrManager Updation\CrManager Updation\Package.dtsx)
* **EQS Data warehouse** (D:\Inout\In\Warehouse\EQS-Data WareHouse-Scheduled\EQS DataWareHouse-Scheduled\EntityChanges\_Invoices\Package.dtsx)
* **Usage Tracking** (D:\Inout\In\UsageTrackingPackage\UsageTrackingPackage\UsageTrackingPackage\Package.dtsx)
* **Set Dates for Turnover** (D:\Inout\In\365 TurnOver\365 TurnOver\365 TurnOver\Package.dtsx)
* **Send Email alert** (C:\Users\ahabib\Desktop\Email Alert\Email Status\Email Status\Email Status\Package.dtsx)
* **Create Email** (C:\Users\ahabib\Desktop\Email Alert\EmailCheck1\EmailCheck1\Package.dtsx)
* **Real time Sync with Cockpit DB (**D:\Inout\In\real time sync with cockpit DB \ real time sync with cockpit DB \Package.dtsx)
* **Rollup fields workaround (**D:\Inout\In\Rollup fields workaround \ Rollup fields workaround \Package.dtsx)
* **Update News Count (**D:\Inout\In\News Count Package \ Companies News Count Package \Package.dtsx) **Update News Count (**D:\Inout\In\News Count Package \ Companies News Count Package \Package.dtsx)
* **Product tile information sync (**D:\Inout\In\Product tile information sync \ Product tile information sync \Package.dtsx)

**Sales Data Scheduled Package:**

* At the start of this package there is a script component which is responsible for checking the presence of a Rechnung file in a source folder and if there is no files in a source folder, then it will send an email notification.
* Next Step is a sequence container which is responsible for checking no of rows in Rechnung file and write it to Empty files table. It is basically to send a notification for empty sales data files.
* Next Step is a sequence container which is responsible for checking the no of rows in Rgpositionen file and write it to empty files table. It is basically to send a notification for empty sales data files.
* Next step is a sequence container which is responsible for checking the no of rows in contract files and write it to Empty files table. It is basically to send a notification for empty sales data files.
* Next Step is a sequence container which is responsible for checking the no of rows in contract line item files and write it to empty files table. It is basically to send a notification for empty files.
* Next Step is a sequence container which is responsible for checking the no of rows in PostenStatus files and write it to empty files table. It is basically to send a notification for empty files.
* Next Step is a data flow task which will extract all of the empty file names from Empty Files table and write them a flat file to be sent as an attachment for empty file names notifications.
* Next step is a data flow task which is responsible for extracting EquityStoryId’s against NAVID1 from CRM and save them to Account NAVID’s table.
* Next step is a data flow task which is responsible for extracting EquityStoryId’s against NAVID2 from CRM and save them to Account NAVID’s table.
* Next step is a data flow task which is responsible for extracting EquityStoryId’s against NAVID3 from CRM and save them to Account NAVID’s table.
* Next step is a data flow task which is responsible for extracting Product UOM’s from CRM and save them to Product table.
* Next Step is a sequence container which is responsible for uploading contract file data into CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* Next Step is a sequence container which is responsible for uploading contract file data into CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* The next step is an Execute SQL Task which is responsible for deleting all of the records from Contracts table and it is using the Table drop and create command.
* The next step is a data flow task, which is responsible for extracting data from CRM for contract number, start date and end date, it will import these exported records into Contract table.
* The next step is a sequence container which is responsible for looping over contract line item files and uploading their data to CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* The Next step is a sequence container which is responsible for looping over Rechnung files and uploading invoices data into CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* As the invoices has been updated, now the next step is a sequence container which is responsible for uploading Rechnung PDF files into share point and link them to their related invoices in CRM. At the very first in this sequence container there is a data flow task for creating folders in share point using script component, in which it will extract the account name of this invoice number and then try to create the new folder in share point with this name, but it will be failed if there is already a folder with this name or directory in share point. Now there are two flows next to it, if it is successful then it will follow the flow of successful direction, in which it will upload the file to that folder and then in the next data flow task it will link this file to a related invoice record in CRM. Upon failure of folder creation, in which it will upload the file or override the file in that folder and then in the next data flow task it will link this file to a related invoice record in CRM. And at the end if it is failed linking it to CRM, then it will move the file to error folder or it will move the file to Archive folder.
* As the invoices has been updated, now the next step is a sequence container which is responsible for uploading Credit Note PDF files into share point and link them to their related invoices in CRM. At the very first in this sequence container there is a data flow task for creating folders in share point using script component, in which it will extract the account name of this invoice number and then try to create the new folder in share point with this name, but it will be failed if there is already a folder with this name or directory in share point. Now there are two flows next to it, if it is successful then it will follow the flow of successful direction, in which it will upload the file to that folder and then in the next data flow task it will link this file to a related invoice record in CRM. Upon failure of folder creation, in which it will upload the file or override the file in that folder and then in the next data flow task it will link this file to a related invoice record in CRM. And at the end if it is failed linking it to CRM, then it will move the file to error folder or it will move the file to Archive folder.
* As the invoices has been updated, now the next step is a sequence container which is responsible for uploading Mahnung PDF files into share point and link them to their related invoices in CRM. At the very first in this sequence container there is a data flow task for creating folders in share point using script component, in which it will extract the account name of this invoice number and then try to create the new folder in share point with this name, but it will be failed if there is already a folder with this name or directory in share point. Now there are two flows next to it, if it is successful then it will follow the flow of successful direction, in which it will upload the file to that folder and then in the next data flow task it will link this file to a related invoice record in CRM. Upon failure of folder creation, in which it will upload the file or override the file in that folder and then in the next data flow task it will link this file to a related invoice record in CRM by writing a link of this file in ss\_reminderurl field. And at the end if it is failed linking it to CRM, then it will move the file to error folder or it will move the file to Archive folder.
* Next step is a sequence container which is responsible for looping over Rgpositionen files and load invoice line item data into CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder.
* Next step is a sequence container which is responsible for looping over Rgpositionen files and to update this invoice line item record’s field “ss\_contractlinelookup”, “ss\_contractnolookup” to link them with the contract and contract line items into CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* Next step is a sequence container which is responsible for looping over reminder files and to update the “ss\_no\_of\_reminders” field of CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* Next Step is a sequence container which is responsible for looping over postenstatus files and to update the status of invoice records in CRM. If there is any error while reading the file or in any case the data flow task responsible for uploading contract data into CRM gets failed, then it will move the file to Error folder or upon successful uploading, it will move the file to Archive folder.
* Next Step is a data flow task which is responsible for copying the missing NAVID’s from Missing NAVID’s table in a file against todays date to send this file as an attachment to a summary notification.
* Next Step is a data flow task which is responsible for copying the missing products from Missing Products Errors table into a file to send this file as an attachment to summary notification.
* Next Step is a data flow task responsible for copying the failed invoice records from RechnungErrors table into a file to send this file as an attachment to summary notification.
* Next step is a data flow task responsible for a copying the failed invoice line item records from RgpositionError table into a file to send this file as an attachment to a summary notification.
* Next Step is a data flow task which is responsible for copying the failed contract records to a file to send this file as an attachment to summary notification.
* Next step is a Send Email task which is responsible for sending summary as an email notification.

**Daily Upload of companies from Cokpit Internal database:**

* In this package there is one sequence container, which have multiple data flow tasks.
* The very first data flow task is responsible for a extracting all users from CRM and load them into Users table.
* The next data flow task is responsible for extracting equitystoryids from CRM and load them into EquityIds table.
* The next data flow task is responsible for extracting companies records on the basis of today’s modification date from Cokpit Internal DB and load them into CRM.
* Then next step is a data flow task which is responsible for extracting companies and their related stock exchange records from cokpit internal DB and load them into ss\_account\_stockexchange entity of CRM.

**Update Cr Manager field of Cokpit DB:**

* In this package there is one sequence container which have multiple data flow tasks.
* The very first data flow task is cache creation, which is responsible for extracting all of the records from cokpit internal DB and write them in a cache file for the purpose of looking up this data later.
* Next step is a data flow task which is responsible for extracting the equityStoryId, ss\_SalesRep IR, ss\_SalesRep Xml from the newly created entity “SalesRep Tracking”of CRM. Then it will follow the condition for updating cr manager and xml\_cr\_manger of Cokpit Internal DB and send the rows, which have satisfied the condition in a Object type variables using record set destinations.
* Next step is to update xml\_cr\_manager field of Cokpit db if sales rep xml is not null using execute sql task.
* Next Step is to update xml\_cr\_manager field of cokpit db if sales rep xml is null by getting the related data from related Object type variable.
* Next Step is to update cr\_manager field of cokpit DB if sales rep ir is not null in CRM by getting the related data from related object type variable.
* Next Step is to update cr\_manager field of cokpit DB if sales rep ir is null in CRM by getting the related data from related object type variable.
* Next Step is to update xml\_cr\_manager field of cokpit db if both the sales rep Ir and sales rep xml are null in CRM by getting the related data from related object type variable.
* Next step is to delete all of the records from SalesRep racking entity of CRM.

**EQS Data Warehouse Scheduled Package:**

* In this package, the data is being synced with local EQS Data warehouse DB from CRM. It has multiple sequence container which are responsible for extracting data from Currency, Users, Teams, Accounts, Contacts, Account Stock Exchanges, Unit Groups, Units, Segments, product group, Price List, Product, Price Line items, Contracts, Contract line items, Invoice, Invoice line items and Opportunity Entities of CRM and load their data to related tables in EQS Data warehouse local DB. We are using EntityChanges feature of CRM to do this sync process
* First Sequence Container is responsible for extracting Currency Entity data from CRM and to load them in to related entity in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting Users Entity data from CRM and to load them in to related entity in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting Teams Entity data from CRM and to load them in to related entity in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting Accounts Entity data from CRM and to load them in to related entity in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting Contacts Entity data from CRM and to load them in to related entity in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting parentcustmerid field of contacts entity data from CRM and to link it with the related account records in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting parentaccountid field of accounts entity data from CRM and to link it with the related account records in EQS Warehouse Database.
* Next Sequence Container is responsible for extracting bbo\_clientagency field of accounts entity data from CRM and to link it with the related account records in EQS Warehouse Database.
* Next Sequence container is responsible for extracting ss\_contact\_account entity of CRM and to load it to related table in EQS data warehouse. There is another thing in it, which is a temporary ss\_contact\_Account table in EQS Data warehouse, it is being used for handling the deleted records because there is no EntityChanges feature for this entity in CRM.
* Next Sequence Container is responsible for extracting ss\_account\_stockexchanges entity records from CRM and to load them into a related table in EQS Data warehouse.
* Next Sequence container is responsible for updating the parent system user id field of users table.
* Next Step is to update the created by and modified by field of Currency table in EQS Data warehouse and to link them to related users or team records.
* Next Step is to update the created by and modified by field of Users table in EQS Data warehouse and to link them to related users or team records.
* Next Step is to extract the records of unit group entity of CRM and to load that data into related table of EQS Data Warehouse.
* Next Step is to extract the records of unit entity of CRM and to load that data into related table of EQS Data Warehouse.
* Next Step is to extract the records of segment entity of CRM and to load that data into related table of EQS Data Warehouse.
* Next Step is to extract the records of product group entity of CRM and to load that data into related table of EQS Data warehouse.
* Next Step is to extract the records of Price List entity of CRM and to load that data into related table of EQS Data warehouse.
* Next Step is to extract the Product entity records of CRM and to load them into related table of EQS Data Warehouse.
* Next Step is to extract the records of Price list items entity of CRM and to load them into related entity of EQS Data warehouse.
* Next Step is to extract the records of Contract entity of CRM and to load them in to related table of EQS Data Warehouse.
* Next Step is to extract the records of Contract line item entity of CRM and to load them to related table of EQS Data Warehouse.
* Next Step is to extract the records of Invoice entity records of CRM and to load them into related table of EQS Data warehouse.
* Next step is to extract the records of Invoice line item entity of CRM and to load them into related table of EQS Data Warehouse.
* Next Step is to extract the records of opportunity entity of CRM and to load them in to related table of EQS Data Warehouse.
* Next Step is to update the related opportunity field of opportunity table of CRM and to link it with the related opportunity record.

**Set Dates for Turnover:**

* This package has 1 data flow task and it is responsible for setting the ss\_turnover\_startdate and ss\_turnover\_enddate fields of invoice entity. Start date is equal to exact 1 year back of today’s date and end date will be equal to today’s date.

**Real time Sync with Cockpit DB:**

* At start of package we are filling up caches (5) in order to lookup from them in package.
* After that we are getting accounts and making filters in Dataflow Task.
* In next DFT we Get opportunities and workflow entity fields and storing them in cache.
* After that there is for each loop Container inside that container there is DFT which Is responsible for creating companies.
* After that DFT there are further for each loop Containers which are responsible for updating of Cockpit Tables.
* At the end there is script task which is used to PAUSE the process for 5 seconds.

**Rollup Fields workaround:**

* At start of package we are storing accounts in cache.
* There is DFT which is storing values in “current year rollup field” in CRM.
* There is DFT which is storing values in “Last year rollup field” in CRM.
* There is DFT which is storing values in “rollup 365 field” in CRM.
* Next DFT is responsible for storing missing customer ID’s Name in flat file.

**Sync xml contacts from cockpit to CRM:**

* At start of package we are storing accounts from crm to cache.
* After that there is DFT which brings contact from cockpit’s “xml contacts” table to CRM by daily basis.

**Update News Count:**

* At start of package we are storing CRM’s accounts data in Cache.
* In next DFT we are getting current year’s “News counts” from cockpit DB and setting them up in CRM’s “ss\_publishednews” entity.
* In next DFT we are getting sum of newscounts from CRM’s “ss\_publishednews” entity and updating crm’s “Account” entity field “ss\_allnewscounts”

**Product tile information sync:**

* At start of package there is sql task which is responsible for drop and create of some tables using in package.
* In next DFT we are storing CRM’s “ss\_accountcontractextra” entity data in Cache.
* In next DFT we are adding CRM’s product groups in DB table.
* Aftrer that there is DFT which is creating that product groups in CRM’s “ss\_ accountcontractextra” entity.
* After that there is CRM’s “currency” data copied in cache.
* In next DFT we are adding invoices from current date to last year’s same date (1 year) from CRM to temp DB table.
* After that we are getting sum of all invoices against respected account from that temp DB table and update CRM’s “ss\_ accountcontractextra” entity’s field “ss\_turnover”.

**Password for all of these packages is “Passw0rd!”.**

<https://equity.officeaccess.de/external_sql>

**username: ahabib**

**Password: c16Q64KGTr**