What is Aranda Field Service?

Aranda Field Service (AFLS) is a tool for the management and solution of work orders, which facilitates the operation of a service, optimizing the processes of automatic or assisted assignment for the fulfillment and effective attention of the field service.

With AFLS, a company that provides services in the field can offer quality service to its customers in three simple steps:

- i) Assignment of orders and programming of activities and resources for timely attention
- ii) coordinated monitoring of the assigned activities and the trips made for the performance of the service
- iii) Attention in the field to the client's request in the agreed times with the timely solution of qualified personnel.

The customer can forget about the operation of their service, the time of attention, the response time and the person in charge since Aranda Field Service will be in charge of automating the service from the allocation of resources to the field attention of the work order, through a powerful assignment engine.

The AFLS location system will allow you to create a competitive advantage by leveraging geographic information components that assist in automating decision-making, lowering operating costs and improving the efficiency of field workers. In this way, mobility and compliance become strategic elements of Aranda Field Service, thanks to the online synchronization of information.

Aranda Field Service has an effective alert system that allows you to automatically identify in real time work orders that present new features that prevent the fulfillment of the service within the established times. Such early warnings are key to timely and effective decision-making.

Aranda Field Service is designed for users and organizations that need to optimize the operation of their services in the field with a timely assignment of cases, improving response times, having real-time access to information, focusing on the service, reducing operating costs, improving the efficiency of specialized personnel and raising the quality of customer service in the designated place.

AFLS Database Considerations

General Recommendations

Objective

To present general recommendations on maintenance and good practices for Aranda FIELD SERVICE AFLS from the point of view of application, Windows services and Aranda FIELD SERVICE database.

Back Up Procedures

Software Back-Up Procedure

On the application server, the web application and Windows services are available. To have a backup copy of this information you must access:

1. On the Aranda server make a backup of the folder: **%ProgramFiles(x86%)\Aranda**.

2. By back-up this folder, you will have the support of:

- Worker and Scheduler Aranda
- AFLS Mapping Engine

3. In the same way, make a copy of the folders:

- **%systemdrive%\inetpub\wwwroot\AFLS**
- **%systemdrive%\inetpub\wwwroot\AssistMe**
- **%systemdrive%\inetpub\wwwroot\ASSISTMEWS**
- **%systemdrive%\inetpub\wwwroot\AFLSAPI**

4 If you have the default storage provider, in both the AFLS and ASSISTMEWS project, you will find the Uploads folder, this folder has the attachments that are uploaded by users related to work orders.

Database Back-Up Procedure

⚐ **Note:** Note that the transaction log-based back-up strategy can increase recovery times; Consider other options such as incremental back-ups according to business needs).

The DBA must select the best strategy to be applied. The following are considerations to keep in mind:
1. Keep the database without user access from Aranda applications and services. Do the following:

- Perform a complete Database Backup periodically. (Define this time according to the importance of the stored data to your business.)
- If you are using full recovery mode[See Recovery Modes in SQL Server documentation], perform a Back-Up of database transaction logs depending on the amount of data that is modified daily and the importance of the data.

Recommendations on Database Operations

If you are running a query or direct operation to the Aranda FIELD SERVICE AFLS tables, you should consider the following recommendations that allow database performance to be uncompromised in the operation:

- You should not manually modify the information stored in the databases, avoiding unexpected behavior in the application.
- Use replicas of the data for historical reporting.
- When building reports, avoid querying all fields in a table using the wildcard character (*)
- When building reports, avoid running SQL commands that query tables in their entirety, instead define WHERE clauses that allow you to get a tight number of rows.
- Don't delete data from tables unless you're clear about the implications. Deleting data may affect the operation of the software. This task may require approval from the audit area and/or the company's management.
- Use the TRUNCATE statement when you need to delete all records from a large table, making the process more efficient since it does not generate entries in the transaction log

Tables with High and Lowest Traffic in AFLS

What are the most trafficked tables in AFLS, where deleting records is not recommended?

- AFLS_WORKORDERS: This table stores the work orders in the system and contains the entire operation of the field work. It currently has different indexes to respond to the most popular searches in the application.
- AFLS_ATTACHMENTS: This table contains the references of work order attachments, customer signatures and SLA attachments; It is not recommended to do any physical debugging as it can generate information integrity problems.
- AFW_ADDITIONAL_FIELD_VALUE This table contains the values of the additional fields of all the concepts of Aranda FIELD SERVICE AFLS (Model, Orders, Services, Customers, Companies, Locations, Products, Web Users and Mobile Users).
- AFW_USERS: This is the main table of users, this table concentrates customers, field specialists and web specialists. This table can be fed new users through the web console, integrations with Aranda SERVICE DESK ASDK, and synchronization from the LDAP active directory. This table should not be purged and its records should not be deleted.

What are the most trafficked tables in AFLS, where records can be deleted?

- AFLS_APP_LOG: This table only has insert operations for each request to create, edit, or delete system configuration settings; its objective is to record the log of operations on the configuration by answering the following criteria: Who made the operation? What concept is involved and what data? What data was used in the

operation?
Note: This table is for audit purposes and it is not recommended to delete your data; If you want to delete records from this table, use the script:

```
DELETE
FROM [BD_NAME].[dbo].[AFLS_APP_LOG]
WHERE alog_generate_date < '2020-01-01 00:00:00'o
```

- AFLS_ASSIGMENT_ENGINE_REQUEST: This table houses the issues that are processed by the Mapping Engine service with their respective solution. This table is supported by an AFLS scheduled task, which deletes by default records older than 2 days.
  If you want to modify the deletion time, you must update the file:
  %ProgramFiles(x86%)\Aranda\Aranda Services\Aranda.AFLS.AssignmentEngineService.exe.config and look for the line:

```
<add key="engine:daystoexpire" value="2" />
```

- AFLS_LOCATIONS: This table groups the geo-referenced points that a specialist records during his work in the field. This table has quite a bit of INSERT operation, since the mobile application, during manual or automatic synchronizations of each specialist, records its location depending on the location accuracy setting (High Accuracy every 10 meters, Medium Accuracy every 30 meters, and Low Accuracy every 90 meters).
  Table AFLS_LOCATIONS is supported by a scheduled AFLS task which passes records older than 3 days to the AFLS_LOCATIONS_HISTORY table; The entire history of field specialists is stored here.
  This table tends to grow a lot, since the more specialists in the field and the greater the availability of hours and days of the week, the more points will be registered. This table can have a periodic physical erase from specific dates, using the following script:

```
DELETE
FROM [BD_NAME].[dbo].[AFLS_LOCATIONS_HISTORY]
WHERE loct_date < '2020-01-01 00:00:00'
```

- AFLS_REVIEWS: This table records the comments added by specialists, dispatchers, end users, and application monitors; as well as automatic comments generated by the system; These reviews help you see "the history" of a work order for changes.
  If you want to debug, it is recommended to check once a month and check if you want to run any debugging.
  Example deletion script:

```
DELETE
FROM [BD_NAME].[dbo].[AFLS_REVIEWS]
WHERE revi_date < '2020-01-01 00:00:00'
```

- AFW_ALERT: This table saves the alerts that the system generates. For example: An order that did not start on time, an inventory product that is about to run out, a satisfaction survey with a low score, etc.; these alerts have 3 statuses (New, Reviewed, and Closed). Depending on the management of the tool, it can generate several alerts and in turn several records.
  A decision can be made to execute a physical deletion script for alerts that have been closed by dispatchers with an effective date since it has an index with the following keys (creation_date, status_id, category_id):

```
DELETE
FROM [BD_NAME].[dbo].[AFW_ALERT]
WHERE status_id = 3 and creation_date < '2020-01-01 00:00:00'
```

The impact of eliminating these alerts in a closed state is to lose track of their management, compared to the tool. For this reason, a deletion and its periodicity should be considered, if necessary.

- AFLS_ALERT_ARCHIVES: Alerts from the table are moved to this table AFW_ALERT of orders that are already in closed or canceled status; these alerts have 3 statuses (New, Reviewed, and Closed). This process is done by a scheduled task that runs at midnight.
  You can make a decision to execute a physical deletion script for alerts that have been closed by dispatchers with an effective date:

```
DELETE
FROM [BD_NAME].[dbo].[AFLS_ALERT_ARCHIVES]
WHERE StatusId = 3 and CreationDate < '2020-01-01 00:00:00'
```

- AFW_ASSISTME_PREREGISTER This table groups the pre-registrations that are carried out from the AssistMe console when it has been enabled from the administration. Pre-registrations mean that end users have requested to enroll in AFLS from AssistMe, once the customer pre-registers, the system sends them an account notification email, to create them as a customer once they successfully validate their account.
  The growth of this table depends on the strategy when using this channel; It is possible that this table can grow by pre-registrations that are pending or have already been enabled. You can make the decision to delete pre-registrations that have not been confirmed for some time or those that have already been confirmed using the following script:

```
DELETE
FROM [BD_NAME].[dbo].[AFW_ASSISTME_PREREGISTER]
WHERE actívate = 1
```

- AFW_MAIL_HISTORY: The AFLS System sends emails to specialists, clients, monitors, etc. for different reasons. These emails are sent by a scheduled task and once they are sent satisfactorily they go to the AFW_MAIL_HISTORY table; This board can grow quite quickly; This history remains intact and a decision can be made under which criteria to delete the information.
  For example, if you want to delete information, you can use the following script:

```
DELETE
FROM [BD_NAME].[dbo].[AFW_MAIL_HISTORY]
WHERE mahi_created < '2020-01-01 00:00:00'
```

- AFW_WORKER_LOG: This table has the task execution log that the Aranda worker service runs. This log is used to identify that the tasks have been executed in the expected time and to validate their status by means of the work_success field (0: Failed, 1: Successful, 2: Pending execution and 3: In process).
  You can make a decision to delete with a certain periodicity the tasks that were successfully executed and leave those that present some type of error in case any support or adjustment is required.

```
DELETE
FROM [BD_NAME].[dbo].[AFW_WORKER_LOG]
WHERE work_execution_date < '2020-01-01 00:00:00' AND work_success = 1
```

Database Maintenance

The maintenance of a database always like that of Aranda FIELDE SERVICE AFLS, is very important to keep all the information updated and at an optimal level. These maintenance operations may be carried out twice a month. It is also possible to modify the frequency of maintenance, taking into account the activity that the database receives in production and the volume of data it handles. This maintenance is recommended to be carried out twice a month, unless the level of transactions is very high.

TRANSACTION LOG:

- Specify a log file growth limit.
- Specify a log file growth rate that prevents a log file expansion operation from exceeding 2 seconds. You can normally use between 30 Mb and 60 Mb.
- If you are using the full recovery model, define a recovery and backup strategy based on the transaction log, to make it easier to reuse the space provided by the log files.
- If you use the simple recovery model, keep in mind that the transaction log will grow slowly, but it could affect the recovery strategy you have defined for the database.

DEFRAGMENTING INDEXES:

Index defragmentation tasks help improve database performance by improving access to data on disk; It is advisable to perform these tasks periodically to ensure the proper functioning of the database. These tasks help the execution plans to be compiled with the latest statistics from the database.

Note: Be sure to measure the impact of third-party scripting operations on your databases. Under no circumstances will Aranda SOFTWARE be held responsible for any damage caused to the information or the database schema after the execution of scripts that have not been referenced by authorized personnel of Aranda SOFTWARE.
If you want to defragment indexes and update statistics, do the following: 1. Make sure no users are editing, querying, or entering data into the Aranda suite tools. You can stop Internet Information Services (IIS) for this.

2. Stop the Aranda Suite services on the application server.

3. Run the script (Index Defragmentation and Statistics Update Script). It is not recommended to update statistics

too frequently as this invalidates the execution plans stored in the server cache. For this reason, in the script you will find the parameter @statistics with the value 'OFF'. When you want to update the statistics, change the mentioned parameter to the value 'ON'.

4. Restart all Aranda Suite and Internet Information Services (IIS) services.

~~~ –Script desfragmentación de índices y actualización de estadísticas use DATABASE –Nombre de la base de datos declare @statistics nvarchar(3) = 'OFF'; –Modifique este valor a 'ON' para actualizar estadísticas. declare @sentence nvarchar(128), @initdate datetime, @inaldate datetime; declare cur_indices cursor for with index_fragment (object_id, index_id, avg_fragmentation_in_percent) as (SELECT object_id, index_id, avg_fragmentation_in_percent FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL, NULL, 'LIMITED') WHERE ((avg_fragmentation_in_percent > 15) OR (avg_page_space_used_in_percent < 60)) AND page_count > 5 AND index_id NOT IN(0) AND object_id NOT IN(OBJECT_ID('AFLS_APP_LOG'), OBJECT_ID('AFLS_ASSIGNMENT_ENGINE_LOG'), OBJECT_ID('AFLS_MAPS_API_USED_LOG'), OBJECT_ID('AFLS_LOG_CRUD_ADDITIONAL_FIELD'), OBJECT_ID('AFW_AUDIT'), OBJECT_ID('AFW_MAIL_HISTORY'), OBJECT_ID('AFW_WORKER_LOG'))) SELECT 'ALTER INDEX ' + ind.[name] + ' ON ' + sc.[name] + '.' + OBJECT_NAME(t.object_id) + ' REBUILD' sentence FROM index_fragment t INNER JOIN sys.indexes ind ON t.object_id = ind.object_id AND t.index_id = ind.index_id INNER JOIN sys.objects ob ON t.object_id = ob.object_id INNER JOIN sys.schemas sc ON sc.schema_id = ob.schema_id Union select 'update statistics ' + OBJECT_NAME(object_id) from index_fragment where @statistics = 'ON' open cur_indices fetch cur_indices into @sentence while(@@FETCH_STATUS = 0) begin set @initdate = GETDATE(); exec sp_executesql @sentence; print @sentence + ' tiempo de ejecución: ' + convert(varchar(10), datediff(millisecond, @initdate, GETDATE())); fetch cur_indices into @sentence; end close cur_indices; deallocate cur_indices; declare cur_indices cursor for with index_fragment (object_id, index_id, avg_fragmentation_in_percent) as (SELECT object_id, index_id, avg_fragmentation_in_percent FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL, NULL, 'LIMITED') WHERE ((avg_fragmentation_in_percent > 10 AND avg_fragmentation_in_percent < 15) OR (avg_page_space_used_in_percent < 75 AND avg_page_space_used_in_percent > 60)) AND page_count > 5 AND dm_db_index_physical_stats.index_id NOT IN (0) AND object_id NOT IN(OBJECT_ID('AFLS_APP_LOG'), OBJECT_ID('AFLS_ASSIGNMENT_ENGINE_LOG'), OBJECT_ID('AFLS_MAPS_API_USED_LOG'), OBJECT_ID('AFLS_LOG_CRUD_ADDITIONAL_FIELD'), OBJECT_ID('AFW_AUDIT'), OBJECT_ID('AFW_MAIL_HISTORY'), OBJECT_ID('AFW_WORKER_LOG'))) SELECT 'ALTER INDEX ' + ind.[name] + ' ON ' + sc.[name] + '.' + OBJECT_NAME(t.object_id) + ' REORGANIZE' sentence FROM index_fragment t INNER JOIN sys.indexes ind ON t.object_id = ind.object_id AND t.index_id = ind.index_id INNER JOIN sys.objects ob ON t.object_id = ob.object_id INNER JOIN sys.schemas sc ON sc.schema_id = ob.schema_id union select 'update statistics ' + OBJECT_NAME(object_id) from index_fragment where @statistics = 'ON' open cur_indices fetch cur_indices into @sentence while(@@FETCH_STATUS = 0) begin set @initdate = GETDATE(); exec sp_executesql @sentence; print @sentence + ' tiempo de ejecución: ' + convert(varchar(10), datediff(millisecond, @initdate, GETDATE())); fetch cur_indices into @sentence; end close cur_indices; deallocate cur_indices; GO

Database Reports

At midnight, a programmed task is executed, responsible for the construction of a data cube by means of the procedure PRC_AFLS_ORDER_STATISTICS which it must be validated that it exists in the database.

Reports in Aranda FIELD SERVICE AFLS are executed through stored procedures. These are constructed when invoked through Aranda's FIELD SERVICE AFLS reporting module.

You can also put together reports from Aranda QUERY MANAGER AQM using the AFLS database as a source; It is advisable to check the periodicity of the execution and the tables involved (i.e. if they are high operation tables) to avoid blockages in operation.

The list of stored procedures that AFLS handles for the generation of reports and dashboard is as follows; You should always make sure that they are created in the database for the correct functioning of the application:

PRC_AFLS_34001_REPORT_DATASET_COMPILANCE	PRC_AFLS_34011_REPORT_SURVEY_DATASET1
PRC_AFLS_34001_REPORT_DATASET_ORDERS	PRC_AFLS_34011_REPORT_SURVEY_DATASET2
PRC_AFLS_34001_REPORT_DATASET_ORDERS_STATE	PRC_AFLS_34011_REPORT_SURVEY_DATASET3
PRC_AFLS_34002_REPORT_DATASET_ORDERS	PRC_AFLS_34011_REPORT_SURVEY_DATASET4
PRC_AFLS_34002_REPORT_DATASET_ORDERS_BY_SPECIALIST	PRC_AFLS_34011_REPORT_SURVEY_DATASET5
~~~

PRC_AFLS_34002_REPORT_DATASET_ORDERS_TIME_BY_SPECIALIST
PRC_AFLS_34003_REPORT_DATASET_ORDERS
PRC_AFLS_34003_REPORT_DATASET_ORDERS_BY_SERVICE
PRC_AFLS_34003_REPORT_ORDER_STATE
PRC_AFLS_34004_REPORT_DATASET_SERVICES
PRC_AFLS_34004_REPORT_TREND_BY_COMPILANCE
PRC_AFLS_34004_REPORT_TREND_BY_SOLUTION
PRC_AFLS_34004_REPORT_TREND_BY_STATUS
PRC_AFLS_34005_REPORT_SOLUTION_TIME
PRC_AFLS_34005_REPORT_WORKORDER_LIST
PRC_AFLS_34006_REPORT_DATASET_COMPANIES
PRC_AFLS_34006_REPORT_DATASET_SERVICES
PRC_AFLS_34006_REPORT_DATASET_TOTALS
PRC_AFLS_34006_REPORT_DATASET_WORKORDER_LIST
PRC_AFLS_34006_REPORT_DATASET_WORKORDER_PROM_COST
PRC_AFLS_34006_REPORT_DATASET_WORKORDER_TOTAL_COST
PRC_AFLS_34007_REPORT_GENERAL_INVENTORY
PRC_AFLS_34007_REPORT_TOP_PRODUCT_FINAL_INVENTORY
PRC_AFLS_34007_REPORT_TOP_PRODUCT_OUTPUTS
PRC_AFLS_34008_REPORT_MOST_USED_LASTMONTH
PRC_AFLS_34008_REPORT_SUMMARY_INVENTORY_BY_LOCATION
PRC_AFLS_34008_REPORT_TOP5_LOWEST_STOCK
PRC_AFLS_34009_REPORT_DATASET_SPECIALISTS
PRC_AFLS_34009_REPORT_MOST_USED_LAST_MONTH
PRC_AFLS_34009_REPORT_SUMMARY_INVENTORY_BY_LOCATION
PRC_AFLS_34009_REPORT_TOP5_LOWEST_STOCK
PRC_AFLS_34010_REPORT_TOP5_MOST_USED_PRODUCT
PRC_AFLS_34010_REPORT_USED_PRODUCT_BY_SERVICE
PRC_AFLS_34011_REPORT_SURVEY_DATASET6
PRC_AFLS_34011_REPORT_SURVEY_DATASET7
PRC_AFLS_34012_REPORT_SURVEY_DATASET1
PRC_AFLS_34012_REPORT_SURVEY_DATASET2
PRC_AFLS_34012_REPORT_SURVEY_DATASET3
PRC_AFLS_34012_REPORT_SURVEY_DATASET4
PRC_AFLS_34012_REPORT_SURVEY_DATASET5
PRC_AFLS_34012_REPORT_SURVEY_DATASET6
PRC_AFLS_34012_REPORT_SURVEY_DATASET7
PRC_AFLS_34014_REPORT_INFO_LOCATIONS
PRC_AFLS_34015_REPORT_INFO_SPECIALISTS
PRC_AFLS_34016_REPORT_INFO_PRODUCTS
PRC_AFLS_DASH_DEMAND_FOR_LABOR
PRC_AFLS_DASH_STATUS
PRC_AFLS_DASH_SUMMARY_SPECIALISTS
PRC_AFLS_ORDER_COST
PRC_AFLS_ORDER_STATISTICS
PRC_AFLS_PROVIDERS_CALCULATE_SCORE
PRC_AFLS_REASSIGN_WO_SPECIALIST
PRC_AFLS_REPORT_DATASET_COMPANY
PRC_AFLS_REPORT_DATASET_LOCATION
PRC_AFLS_REPORT_DATASET_PROVIDERS
PRC_AFLS_REPORT_DATASET_SERVICE
PRC_AFLS_REPORT_DATASET_SURVEY_NAME
PRC_AFLS_REPORT_DATASET_SURVEYS
PRC_AFLS_TRANSFER_INFO_LOCATION_HISTORY
PRC_AFLS_WORFLOW_ACTION_CHRONOMETER
PRC_AFLS_WORFLOW_ACTION_COST

⚐ **Note:** It is recommended to check the latency between the application server and the database server, check the network hops to get the data.

References:

- SQL Server database maintenance
– Aranda Support Group.
- M2PAS – Database Standard.