Agentry and SMP Metadata Performance Tuning

Guidelines for performance tuning with Agentry and SAP Mobile Platform Metadata based applications

TABLE OF CONTENTS

This document provides assistance on improving Agentry application performance. It lists various challenging scenarios and provides suggestions on how to alleviate them. This is followed by a list of tips for improving application performance. Because of the disparate nature of each customer's environment, not all suggestions and tips will be useful in all cases. Performance Tuning goes hand in hand with Performance Testing and the Agentry and SMP Metadata Performance Testing may be useful in that regard. The information in this document was primarily tested with Agentry 6.0.x but the information is expected to still be relevant to the Agentry and Metadata component in SMP 2.3 and 3.0.

CLIENT INITIAL LOAD TIME

The client takes a long time to do the initial load of the application logic and data.

Reduce Overall Sync Time

The first thing to consider should be to try and reduce the overall sync time because this provides benefits to all stages of the application. This consists of reducing Client Login, Complex Table, Object Fetch and Transaction Submission Time as mentioned below.

Pre-built client

In many cases master data like functional locations, equipment, etc. end up containing the same data for a significant proportion of the end-users. In this case, it becomes viable to pre-build a production client using a generic user and deploy the pre-built client to the end-user devices. The deployment of the pre-built can also be combined with the branding tool to provide a branded installer with ready data. The pre-built client deployment can continue to be managed by the same tools that would be used to manage the Agentry Client.
The benefits of this solution are firstly that the end-user does not have to wait for the complete application logic or data set to download over the data network, thus leading to a much lower initial sync time. Secondly, the memory and resource usage on both the Agentry Server and the ECC backend are reduced because they are now serving a smaller volume of data.
In the case, the customer has separate region with separate sets of master data, a pre-built client would have to be created for each region. If the customer does not wish to deploy a fully pre-built client, the pre-built can be modified so that only certain tables are included or any other such combination.

CLIENT LOGIN AND SYNC TIME

Reduce Backend Sync Latency

Analysis of server logs will reveal the total time it takes for a request such as the login request to be processed and returned. Latency in frequently performed requests adds up and will lead to reduced responsiveness. It is useful to consider and review the following:

- Agentry Java processing time
- Time for request from Agentry Java layer to be sent to and be processed by the backend
- BAPI processing time, including warnings and messages
- Network latency and distance between Agentry Server and ERP system(s)

Improve Device performance characteristics

Modifying the device configuration can sometimes have significant benefits. This can include:

- Updating the device to a newer firmware version
- Changing the faulty device driver of a specific component

- Adding more memory
- Replacing the storage medium (e.g. SD Card) with a higher performing model. A faster external SD card can sometimes outperform the internal storage.
- Modifying OS settings. For example setting ""No remove on resume", SDMMC instead of SDIO, turning write caching on or off, setting display setting to max performance, etc.
- Modifying Anti-virus to exclude Agentry subdirectories from "check on modify"

COMPLEX TABLE SYNC

Reduce number of rows

Review the data required by each user for each scenario and reduce the data to only the records required by the user.

Reduce number of columns and indexes

Deleting columns and indexes reduces both the data storage and transmission volume and the processing time for the indexed. The Editor's dependency viewer can be used to check where a Complex table column or index is being used. Once a column is deleted from the Agentry editor, the column should also be removed from the java and from abap both for consistency and performance.
In Java, this can be done by extending the base class and overriding the public classes to private and removing the value population code.
In ABAP, this would be done by using the Agentry Config Panel.

Optimize column length

Review the backend data structure to ensure the complex table column description and length are accurate. String fields should not be longer than their backend equivalents. This keeps the application validation tight and improves performance.
Consider the following when sizing these fields in regard to the storage requirements of these values.

ASCII String

There are two ASCII string types available for fields within a complex table: case sensitive and case insensitive. Case sensitivity affects sorting and searching behaviors on the Client and does not impact storage requirements.

- For both ASCII string types, the developer must specify a maximum length that represents the maximum number of characters the field can contain. This maximum length determines the amount of storage required for that field in all records, regardless of the actual length of the string in a given field.
- ASCII strings are automatically padded to the next even number. Therefore, if you have an ASCII string with an odd length, an extra byte will be added to make it even.
- For ASCII strings, one character equals one byte. An ASCII string field with a maximum length of 10 characters requires 10 bytes of storage per record. Therefore, a complex table with 1,000 records would require 10,000 bytes for this field.

International String

There are two International string types available for fields within a complex table: case sensitive and case insensitive. Case sensitivity affects sorting and searching behaviors on the Client and does not impact storage requirements.

- For both International string types, the developer must specify a maximum length that represents the maximum number of characters the field can contain. This maximum length determines the amount of storage required for that field in all records, regardless of the actual length of the string in a given field.
- For International strings, one character equals two bytes. An International string field with a maximum length of 10 characters requires 20 bytes of storage per record. Therefore, a complex table with 1,000 records would require 20,000 bytes for this field

Identifier

The identifier field type is used to store the value that uniquely identifies each record in the table.

- This value is stored as a numeric value of type integer.

- Values must be greater than 0, no negative or zero values.
- An Identifier field requires 4 bytes per record.

Number

The Number field type is used to store numeric values.

- This is a 32-bit value.
- A Number field requires 4 bytes of storage per record. For example, a complex table with 1,000 records would require 4,000 bytes for this field.

Combine tables

Sometimes it may be useful to consider combining two or more tables into one, if they are usually used in conjunction. The Java layer can also be used to do this without making any ABAP changes.

OBJECT FETCH TIME

Similar recommendations apply to objects.

Reduce number of objects in a collection

Reducing number of objects downloaded will reduce the download time. Also reducing the number of objects in a collection will reduce the seek time for any rules searching within the collection and the list display time.

Reduce number of properties

Reducing number of object properties will reduce the download time. In this case, non-populating the value in Java or ABAP will also reduce the download time (by a smaller amount) even if the property exists in the Agentry object.

Combining fetches and objects

Reducing the number of fetches will also improve the sync time because the number of round trips are reduced. This can be done by reducing the number of object or collections

Use sub-collections

If only a few properties on an object are constantly frequently updating (e.g. status) while the rest remain static, consider separating those values into a sub-collection. This way you can send a smaller amount to data (or even use push) instead of sending the whole set every time

TRANSACTION SUBMISSION

Transactions are how data is modified by the Agentry Client and the Apply step in an action causes the modified data to be written to the physical storage. Transaction with Update Steps are also how data is created and packages to be sent to the Agentry server for processing. Slow transaction processing does not affect initial load but can affect regular transmits and client responsiveness if there are a lot of transactions waiting to be processed. Following are some techniques for overcoming that.

Only include the required properties

Include only the properties that are being modified or are essential to the transactions workings.

Use Apply less frequently

If an action is initiating a loop or other sub-action(s) then performing the Apply at the end of the parent action instead of the sub-action is much faster. However care should be taken to ensure that the flow will always lead to that Apply step. This can be quite useful for add, modify and delete loops.

Combine Update Steps and transactions
Each update step causes a round trip to the backend, combining the update steps (and transactions if possible) will reduce the round trip time for significant time savings. Combining transactions even without update steps will also yield benefits.

Send a collection to the backend instead of multiple individual objects
This takes the previous suggestion even further. If the application flow indicates that a lot of updates to the same object type will be posted to the backend at the same time (e.g. inventory count), then combining them may be useful. A transaction with the collection of objects as its property may be used to store all the objects. This transaction can then send all the values to the backend in a single update step.

APPLICATION FRONT END RESPONSIVENESS

The Agentry Client displays screen with data and processes actions that take the user from one screen to another. A natural transition time from one screen to the next provides a good user experience, below are some scenarios that may negatively effect user experience and suggestions for resolution.

List scrolling too slowly
The number of items in the list may be very large, try setting up filters and include rules to reduce the number of displayed items. You can also add a button and action to work with the include rule to provide more flexible functionality.
The format rules on some columns may be taking too long to execute. Precalculating the values and storing them in properties will speed up the display.

Detail Screen displaying too slowly
When a detail screen is shown, all the tabs in the screenset also have their values calculated (and rules executed). If one those tabs is showing a lot of data or has a lot of complex rules it will slow down all the other tabs. Considering putting that tab in a separate screenset.

Transaction Screen displaying or updating too slowly
When a transaction screen is shown, all the update rules are being calculated whenever values are updated. Consider pre-calculating some values and putting them in transaction properties or hidden fields on the screen.

Next step after a button press too slow / Slow Action execution
Check the application flow in debug mode to ensure that the sub-action branches, loops and rules are efficient. Pay particular attention to rules traversing sub-collections.

AGENTRY SERVER PROCESSING TIME

Optimizing Java Heap Usage
See Admin Guide on how to monitor Java Heap Memory Usage. After running performance tests, it should be possible to find the range of java memory usage. Modify the Agentry.ini Java sections values for initialHeapSize and MaxHeapSize. The initialHeapSize should be the average of the max and min usage values and the MaxHeapSize should be slightly higher than the max usage. This reduces the chances of the server having to ask for memory after it has started/

Use Java Server JVM
By default the Java VM used the Client VM. Using the Server VM will provide better performance. The server vm is usually located in the server subdirectory of the JRE folder and can be set by modifying the PATH or JAVAPATH.

www.sap.com