



ebook:

Hybrid copying and masking of SAP SuccessFactors data

Paul Hammersley, SVP of ALM Products
at EPI-USE Labs



What's the best way to manage data in your complex HXM environment?

Clients in hybrid scenarios (SAP ECC On-Premise and SAP SuccessFactors Employee Central) – or those using SAP SuccessFactors Employee Central Payroll – have limited options for making test data available across these environments.

You are also likely to need access to your critical HCM and Payroll production data in non-production systems for testing, training and support, but in doing so you also then require the consistent masking of SuccessFactors data. Even after an Instance Refresh, you must still anonymize the data to comply with SAP's data-processing agreement.

SuccessFactors doesn't offer the same transport mechanisms as traditional ERP systems. Keeping your ERP, Payroll and SuccessFactors systems consistent for testing is a challenge. **Data Sync Manager Object Sync for SuccessFactors Add-on** offers the ability to intelligently manage your data within the landscape to increase efficiency.

Masking of data consistently between your ERP, Payroll and SuccessFactors systems is not something any standard tools can deliver. **Data Sync Manager Data Secure for SuccessFactors Add-on** offers the ability to mask data intelligently and consistently across existing ERP/Payroll and SuccessFactors systems.

Read on to find out how these solutions were developed and how they will work in your environment.

Genesis

The exact birth of our Data Sync Manager™ (DSM) software is quite a disputed topic with many colleagues who have greater tenure at EPI-USE than I do. To put that in context, if one of my children had been born the day I joined EPI-USE, they would now be an adult! If pushed, I would suggest the most accurate lineage of DSM would place it as being 25 years old in 2024. What was originally known as DSM we now refer to as 'Object Sync HCM'. The 'data on demand' ability to copy specific instances of a data type grew beyond just HCM. To enable that with DSM version 3, an area called the Business Object Workbench (BOW) was created which allowed our software engineers to define 'Objects' that the engine could then consume to read on the source and create/update on the target. This includes many types of information about the object including Selection criteria, Authorisation checks, Lock objects and many more. The most prominent information stored in the BOW was the SAP tables used to store data for an instance of the object, and how those tables related to each other to allow the copying of data for a specific instance. The scope of DSM then expanded further over the years to cover the needs of more technical teams, such as lean test system refresh, new system shell creation and in-place masking of existing data/clients.

Despite over 3000 objects (across S/4, CRM, SRM, GTS, BW etc etc) now being defined in the BOW, and the significant value many multinational companies get from lean landscape management, the ability to copy a set of employees quickly and accurately with their full payroll and time history is still one of the most valuable, easy to understand, and easy to execute functionalities. This is because of two main reasons:

The HCM data model is so different to other areas, and everything needed can easily be encapsulated under one employee number. In FI/LO there are more dependencies between a multitude of objects, which mean more thought and consideration is deployed in copying data on demand.

The employee data changes more frequently and has to be up-to-date for accurate testing, particularly in the areas of payroll and time. A Material Master or Business Partner may not change at all between test system refreshes, but an employee will get new payroll results every month or sooner. And the accuracy of the next payroll run is dependent on the values in the previous ones.

And of course, with the deep domain knowledge in our organisation, we could also effectively anonymise identifiers like name, address, government ID, etc. as the data was being copied, so it was 100% accurate for testing issues, but of zero use to a hacker or other nefarious actor in the test system where people often have a lot more permissions.



The topology of SAP SuccessFactors



The reason DSM was so effective at reproducing the data of an employee, and often therefore the issue that had been logged in Production in a test system, was because of the ability to interact directly with the underlying database. Unpacking the clustered data into its logical tables, adapting values that needed to be changed and then repacking it to the format needed to update cluster tables on the target system.

SAP specifically bought SuccessFactors because it was a 'born in the cloud' or 'pure cloud' solution where multiple organisations can run business applications and processes across a shared set of hardware and database(s). Until I started working with SuccessFactors, I naively thought that meant that the solution was far more templatised, with organisations leveraging best-of-breed business processes they would not feel the need to deviate from. Without the ability to deploy code directly on the database, that is in some parts true; but in terms of the data model and business rules, SuccessFactors allows a multiplicity of ways to customise the solution.

The mechanisms for reading and writing to the system, though, are very different. From within the User Interface (UI), a super user can download data as .csv, and in many cases is able to upload data from files too. There are defined integration mechanisms for the replication of data with an Employee Central Payroll (ECP) or on-premises SAP system for payroll, time etc. But for our requirements of replicating Production data in a test instance, the only show in town is APIs.



There are two available:

The older one, and less preferred for client integrations, is the SFAPI, which is a SOAP type API that is now deprecated, with exception of the Compound Employee. The Compound Employee is used to replicate employee master data from Employee Central to SAP systems, payroll systems, and benefits systems. SAP has advised that all new API development be done using the OData technology, and also to move any legacy applications and integrations to the OData APIs.

The more widely usable one is the OData API which is a RESTful API that has a slightly adapted implementation by SuccessFactors compared to the generic OData standards.

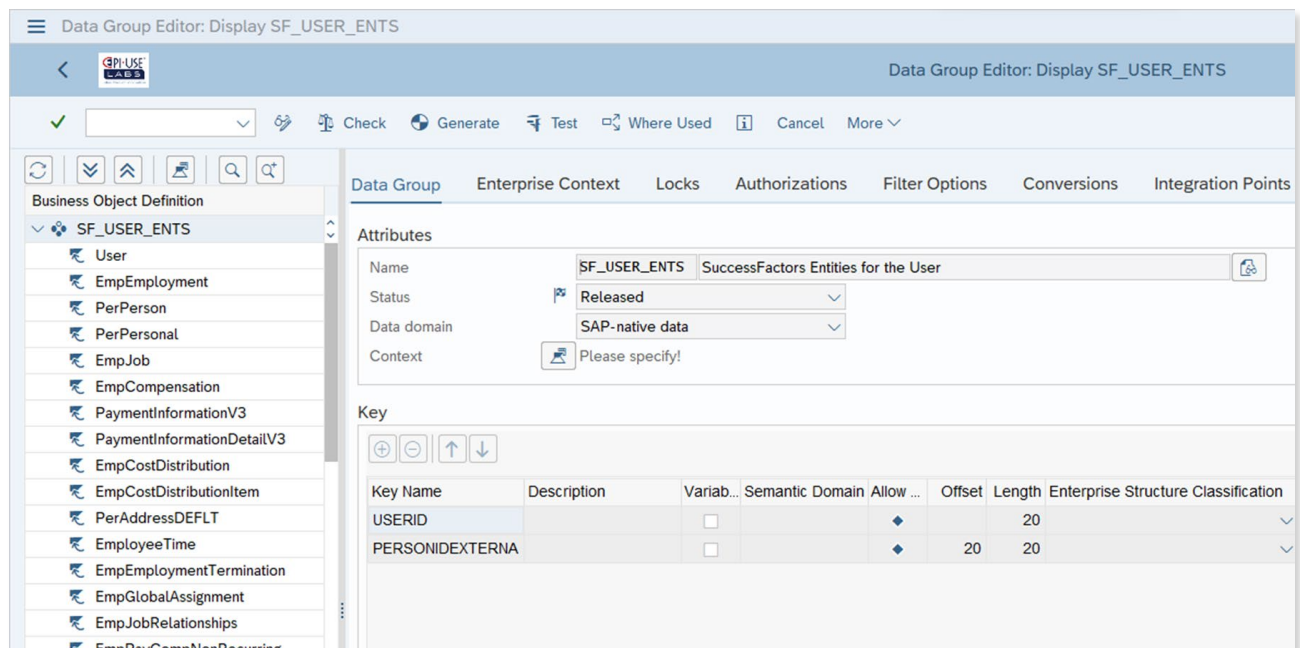
Check out [OData.org](https://odata.org) for more information about the API, and there is also a [useful SAP blog](#).

When we expanded DSM from HCM to all areas of ERP, we realised that for most transactional data we needed to leverage a BAPI (Business Application Processing Interface) to ensure all the configuration of the system and related functional areas was correctly updated – such as validating ‘Availability to promise’ before an order is created, or updating controlling correctly when a finance document posts. This meant a different expectation from copying employees, where 99.9% of the time a user could just enter the relevant employee numbers, hit execute and then start testing with their data. The user needed to be more au fait with the techno/functional aspects of the system to understand why an order wouldn’t create if there wasn’t stock, or to move the posting periods on. But it also meant that DSM was providing validation of the data in the test systems. In some cases, the reason the BAPI wasn’t successful was because of a config change on its way to Production, and the failed syncing of Production data was an early warning that prevented that change going live as-is and causing a Production outage of a key business process.

The OData API similarly carries out validation against the data being copied, and indeed anonymisation values, that simply doesn’t happen on the traditional ABAP side. More on that later...

Integration of OData

In order to properly support the copying and anonymisation of SuccessFactors data, a fundamental change was required to the Business Object Workbench. It now needed to be able to store information about OData entities. Whereas tables had fields, entities had properties. The mappings of fields that stated how tables were joined need 'Nav' routes to be available instead. This was done with the same principles of extensibility, so that the exact definition of the OData model for a particular SuccessFactors instance could be modelled automatically from the meta-data, and to allow the client to add their own custom entities to the model as an extension.

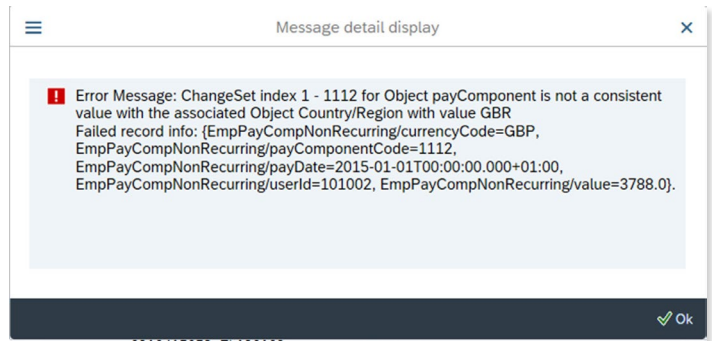


Once the entities could be mapped and stored, then the properties of these could be made available to Data Secure™ (part of the Data Sync Manager software suite). This meant the Integrity Maps that joined table fields with the same semantic values would now need to be able to include OData entity/property values and the anonymisation engine able to access them.

Rule builder		
Option: Mask email address and remove all other cor		
PA0105_USRID_LONG	EMAIL_FR_NAME	Last Name: PA0002_NACHN; First Name: PA0002_VORNA
PA0105_SUBTY	DELETE	
PA0105_USRID_1	DELETE	
PEREMAIL_EMAILADDR	CONSTANT	Constant value: PA0105_USRID_LONG
USER_EMAIL	CONSTANT	Constant value: PA0105_USRID_LONG
Option: Mask email address and remove all other sut		
PA0105_USRID_LONG	EMAIL_FR_NAME	Last Name: PA0002_NACHN; First Name: PA0002_VORNA
PA0105_SUBTY	DELETE	
PA0105_USRID_1	DO_NOTHING	
PEREMAIL_EMAILADDR	CONSTANT	Constant value: PA0105_USRID_LONG
USER_EMAIL	CONSTANT	Constant value: PA0105_USRID_LONG
Option: Set email to constant value, remove other sul		
PA0105_USRID_LONG	CONSTANT	New email address: scrambled@email.etc
PA0105_SUBTY	DELETE	
PA0105_USRID_1	DELETE	
PEREMAIL_EMAILADDR	CONSTANT	Constant value: PA0105_USRID_LONG
USER_EMAIL	CONSTANT	Constant value: PA0105_USRID_LONG

I'm just not haPI

The ability to store OData definitions and leverage that to build requests to read data on the source, anonymise the data consistently and then create upsert requests on the target was a great step forward, but inevitably doesn't bring the same outcome as being able to read directly from and update a target system's database. There are valid reasons why the API will fail, and these fall into three main categories:



A misalignment between the configuration of the source and target system.

SuccessFactors was initially designed with the mantra that business users could make changes in the system behaviour themselves. There is no 'Transport Management System' akin to the ABAP stack one....yet. [See this white paper for more details](#). So you can often have, for example, a picklist option that exists in Production, but not in a Test instance. How should an API call resolve that automatically?



A misalignment between the data and the configuration.

Within the API dictionary, fields can be deemed 'Required' meaning that an upsert for that entity must have a value for that field. In an ABAP system, such a setting would require that the field is populated with a value for any existing data (normally when someone tries to maintain it, the screen will not allow them to save until they've populated any mandatory fields). But in SuccessFactors the property can happily remain empty for instances of the entity, even when the API has it as 'Required'.



A configuration reason why historical data cannot be processed.

An example of this would be a business rule that checks the start date of a compensation record and raises an error if it's older than a certain date. In normal productive use of the API, it would make perfect sense to prevent fraud, but in the copying of historical data to a test instance, it obviously causes problems. In this situation you can disable all Trigger Rules for the object in 'Manage Business Configuration' on the target instance or alternatively place an 'IF' statement at the top of the Business Rule that checks the user and does not execute the logic if it's the API user calling (assuming Basic Authentication is used).

Masking or data on-demand



At the very outset, I would ask someone interested in our Hybrid add-ons what the main driver is for their organisation:

- Do they have a requirement to anonymise data after an instance refresh?
- Or is there a need to copy data on demand for testing, training or defect resolution?

If it's the former, then I would recommend our Data Secure solution because this negates the first type of API error. There can still be issues of type 2 and 3 underlying, but some of these may be irrelevant if the entity does not form part of the anonymisation policy. Only the entities which are to be changed will be sent in the upserts.

Object Sync™ (part of the Data Sync Manager software suite) will simultaneously anonymise data consistently between Production and test, and the Policy defined can optionally be available in Data Secure as a standalone too. But the main driver here would be enabling techno/functional users to sync data themselves on demand, and ensuring that real sensitive data is not being made available in instances which are less tightly policed.

A requirement for the successful use of Object Sync is that the target instance has at some point been a copy of the source via an instance refresh. If this is not the case, there will be too many misalignments of existing data, for example, a Position code on the target is already in use but in a completely different part of the organisation. Or a User/ PersonIdExternal combination is blocked because one or other already exists in the target instance used with a different User/PersonIdExternal (this means an EmpEmployment record that cannot be replaced via OData calls because both fields are business keys in the OData dictionary).

Sometimes there can be additional foundation data that is new and is also required in order to allow the User record to be created or updated via the API. This is why the SuccessFactors Alignment Check utility was created.

THE utility

The SuccessFactors Alignment Check rather undersells what this can do for expert users. It has a couple of comparison reports that show Meta-data or Picklist differences between source and target instances. This is very useful for troubleshooting, but it is also a good way to explain the change management issue that many organisations stumble into with SuccessFactors. Often a new property or picklist type or item is created in one test instance and then again in the Production instance. Some organisations have a QA instance as well, or a Preview instance where the change isn't made because the person making the change didn't need to test there as well. Sometimes, as I mentioned earlier, changes are made directly in Production and nowhere else! Over time the instances being tested on have become unrepresentative of the Production instance, which invariably damages the value of the testing being done.

The ability to read and compare meta-data is required for the Object Sync engine to try to avoid basic misalignments like a property not existing on the target. But both Object Sync and Data Secure are essentially ways of automatically building and submitting OData requests, with the entities that are needed, and the anonymisation routines, packaged together so the user can work at a level above; in other words, selecting the pernr values they need to test/mask and the product taking care of what that actually means in terms of OData calls. In implementing Object Sync the first few times we hit so many situations where we wanted a more fine-grained approach. For example, a single entity is failing to upsert and is preventing dependent entities from upserting. Here we wanted to not only send that entity in isolation but potentially even change or clear a particular property to work around an API issue.



SuccessFactors Alignment Check

Step forward 'THE' utility. Any entity in the meta-data can be selected and a dynamic selection option is created based on the key fields. Nav properties can be pre-selected before the first OData read from the source, or when the data has been retrieved, expands on Navs can be carried out and a follow-on call is made to read the data just for that particular main entity selection. From there, the values for Source and Target instances are visible and can be compared. Missing properties in either are shown as such. And the user can amend values on any entity which is upsertable. Then, if they have permissions (there is a separate auth check for this) and if the target instance is not a Production one, they can choose to Upsert their selections. This prompts a pop-up where the order in which entities are sent can be chosen, so dependencies between Position and FODepartment, for example, can be catered for, and the user can decide whether the OData call should be made with the 'Full purge' option or not.

If you are an existing client of our solutions, take a look at [this video](#) which showcases using the utility to copy missing pick list entries.

Cloning vs Copying

Cloning employees is far more complex on the ABAP side than you might realise. Considering Concurrent or Multiple Employment, decisions about whether to clone Positions in the org structure or have multiple incumbents in a position, are not trivial. When you extend the Central Person, Employee number, Position to now also take in User, PersonIdExternal and a multitude of other visible and hidden identifiers in SuccessFactors, cloning is simply not realistic. The mess of data it would create would also be nigh on impossible to clean up, with no capability to fully purge Users and all their related data from the OData API.

So, with Object Sync SuccessFactors we don't permit the cloning or renumbering of employees and their related Users because of the potential detrimental effect on SuccessFactors. The recommendation is to use an Instance Refresh to regularly align test instances with Production, optionally (although it really isn't if you read your SAP Data Processing Agreement) mask the data with Data Secure, then update existing employees or bring down new hires with Object Sync until the next instance refresh.

Recommendations

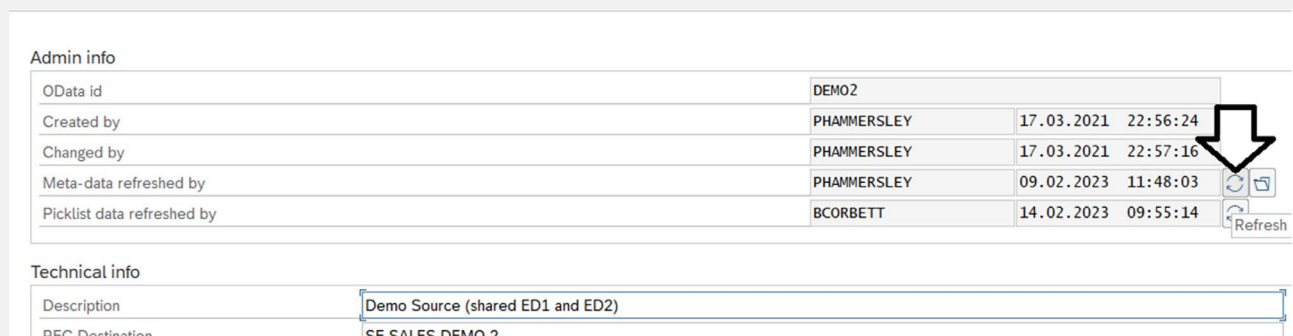
Hopefully, understanding more about how the technology works will better enable you to make the best use of it. Whereas DSM Object Sync HCM could be leveraged by a wider user group to accurately recreate testing and training examples, DSM Object Sync for SuccessFactors Add-on, and the Add-on for Data Secure, perhaps require a more technically savvy user who has knowledge of the data model and can interpret what is really an issue from the OData messages that are received.


For organisations that do not have those skills in-house, this is something our professional services organisation can also assist with.

My top tips are as follows:

1. Check the Meta-data is up to date

On the SuccessFactors side, go to "Admin Center" > "OData API Metadata Refresh And Export" and then click "Refresh". Then inside DSM Administration -> Control Center->OData identities, go into your particular OData identity in change mode and click on Refresh Meta-data:



Admin info				
OData id	DEMO2			
Created by	PHAMMERSLEY	17.03.2021	22:56:24	
Changed by	PHAMMERSLEY	17.03.2021	22:57:16	
Meta-data refreshed by	PHAMMERSLEY	09.02.2023	11:48:03	
Picklist data refreshed by	BCORBETT	14.02.2023	09:55:14	
Refresh				
Technical info				
Description	Demo Source (shared ED1 and ED2)			
PEC Destination	SE SALES DEMO 2			

You can also refresh the picklist data at the same time (this happens automatically during a run if it's out of date though).

2. Does this scenario actually require SuccessFactors data?

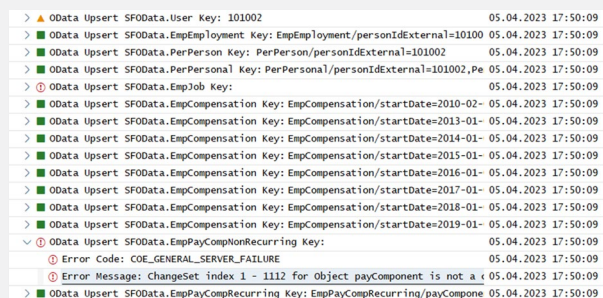
We initially set it that the 'SuccessFactors User' datagroup would be on by default if the system is licensed for Object Sync SuccessFactors. We recently reversed this decision because there are many situations where you wish to test on the payroll side and simply don't need to update SuccessFactors. This could also include situations where you wish to clone or renumber employee numbers into a different number range. The only downside is SuccessFactors scrambled values will not match the backend scrambled values, but the solution is to simply copy them again with SuccessFactors data included when it becomes an issue.

3. Not every red light is a stop sign

Which entities need to successfully upsert depends on:

- your testing scenario/requirement
- whether the data has changed for the employee since the last instance refresh
- whether the entity includes data that is in the masking policy.

In many cases, the data for the entity that is failing is already the same data in the target instance. So, often there is no requirement to even investigate or do anything. See the point below to verify if the data is already correct.



> ▲ OData Upsert SF0Data.User Key: 101002	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpEmployment Key: EmpEmployment/personIdExternal=10100	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.PerPerson Key: PerPerson/personIdExternal=101002	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.PerPersonal Key: PerPersonal/personIdExternal=101002,Pe	05.04.2023 17:50:09
> ⚠ OData Upsert SF0Data.EmpJob Key:	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2010-02-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2013-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2014-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2015-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2016-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2017-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2018-01-	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpCompensation Key: EmpCompensation/startDate=2019-01-	05.04.2023 17:50:09
> ✓ OData Upsert SF0Data.EmpPayCompNonRecurring Key:	05.04.2023 17:50:09
> ⚠ Error Code: COE_GENERAL_SERVER_FAILURE	05.04.2023 17:50:09
> ⚠ Error Message: ChangeSet index 1 - 1112 for Object payComponent is not a	05.04.2023 17:50:09
> ■ OData Upsert SF0Data.EmpPayCompRecurring Key: EmpPayCompRecurring/payCompone	05.04.2023 17:50:09

4. What does the issue look like with the utility?

When Object Sync (or Data Secure) updates the data on the target instance, they are submitting a number of entities in a specific order. Sometimes it's very instructive to take an entity that's failing and use the SuccessFactors Alignment Check to attempt to send specific entities on their own. Just viewing the data from source and target side-by-side reveals the cause of the issue. When it doesn't, then being able to send an upsert but change a property value or control the purge options can help. Or it could be that a related entity needs to be sent but the order in which they're sent is crucial. As an existing user, please log a support ticket on Client Central if this happens, so we can see if it's a standard issue that can affect everyone. If it's an MDF entity, we may need to assist with changing the way it's linked as an extension in the Business Object Workbench.

5. Working around required properties being empty

In the situation where you cannot upsert an entity for a User because there is a property which is 'Required' but not populated, I would recommend the following order of consideration:

- 1) Does the business want to populate the property for existing data in the Production instance?
- 2) Does the property need to be 'Required' in the OData API dictionary? Could the same outcome be achieved via a Business Rule which could be set as not active for API calls?
- 3) If none of the above are possible, then we can create an exit in DSM which can use logic to decide how to populate the property. Look for any potential way of deriving a value and what the impact might be for someone trying to test with the data in the future.

Troubleshooting options

There are three Advanced options within Object Sync that can help when troubleshooting SuccessFactors issues:

Advanced options

- ☐ Only insert SAP UserID Communications Infotype if it is unique
- ☐ Clear TIME link to Production Order Confirmation
- ☐ Remove Infotype Text
- ☐ SuccessFactors: enable OData single insert for Troubleshooting 1
- ☐ SuccessFactors: allow Position multiple incumbents 2
- ☐ SuccessFactors: for Upsert, clear property values if they are invalid Picklist options 3

1. SuccessFactors: enable OData single insert for Troubleshooting

If you are escalating issues to us via a Diagnostic Log and a ticket, you could repeat the run with the option 'SuccessFactors OData single insert for Troubleshooting' checked so we can see individual responses to each entity rather than a batch submission response covering all entities of that type for that person.

2. SuccessFactors: allow Position multiple incumbents

If the Position you're attempting to send a User to is already in use, you can:

- a) see where that User is in Production and potentially copy them as well to free up the Position your original User should go to
- b) manually move the loiterer in the Position on the target instance
- c) manually configure the Position in SF to allow multiple incumbents
- d) tick this option that allows DSM to do step C for you automatically.

3. SuccessFactors: for Upsert, clear property values if they are invalid Picklist options

When a property uses a PicklistV1 navigation the OData API is expected to provide the internal code for the value stored. DSM keeps the picklist information from the source and target instance and carries out an automatic conversion of the value from the internal code to the external value in the source, then on the target it looks up the internal code used there via this external code.

In the event of the value not being found in the Picklist data the DSM engine will send the external value, as it would have done if the navigation used the Picklist V2 approach. There are some entities such as PerAddressDFLT where this is required to avoid an incorrect regional Picklist being used for validation of the values.

This tick box instructs the engine to clear properties that are not 'Required' if they are not found in the Picklist data. This may allow something to post that otherwise would not have done but keep in mind – this applies to all the Entities DSM is sending for a User. A better approach to resolve a specific Entity issue might be to use the utility described earlier to process only the problematic Entity for a User and manually clear the offending property. The rest of the User record having already been updated by DSM.

Exclusion table /USE/OS3_PROPEXC

Earlier, we covered situations where configuration or historical data simply does not allow an Entity to be upserted via the OData API because of issues with specific properties. DSM has an exclusion table where you can provide the Entity/Property combination and set that it should be excluded from the payload. It applies to Object Sync, Data Secure and the SF Utility. From Build 222 onwards, this is maintained via the Control option in the DSM Admin Launchpad.

In summary

The SAP HCM/HXM space has been redefined over and over in the last decade. For many organisations, the current and future state involves a hybrid between ECP or an on-premises payroll or S/4 system with Employee Central. This presents a myriad of test data and privacy challenges which span very different technologies.

In extending DSM into the SAP SuccessFactors area, we had no idea of the challenge that lay ahead for us and for our clients. As that world has unfolded, we've looked at how to build capabilities to allow the user to resolve or work around some of the complexities. This may mean that a slightly more technical skill-set is needed to achieve the same aims, compared with cases where only the ABAP stack SAP system was in play. It may also mean in each organisation there is a wider group of users, and a much smaller group of super users that support them, leveraging DSM in this hybrid way.

Hopefully, this ebook gives the super users information they need to get the best of DSM's different functionality and achieve their own objectives, and to problem-solve for their user community; or helps a user to make the transition to become a super user who can help others.

As a global software solutions and managed services company, EPI-USE Labs helps you to maximise the performance, management and security of your SAP and SAP SuccessFactors systems. Our clients tell us every day how we have transformed their business operations. Contact us to find out how we can help you solve your business challenges.

EPI-USE Labs is a member of groupelephant.com.



groupelephant.com is a largely employee-owned group of companies, nonprofits and impact investment organizations, with a strong global presence. The Group is characterized by a primary strategic imperative in terms of which it goes 'Beyond Corporate Purpose' in its day-to-day activities.



www.epiuselabs.com



clientcentral.io



EPI-USE Labs



sales@labs.epiuse.com



[@EPIUSELabs](https://twitter.com/EPIUSELabs)



[EPI-USE_Labs](https://www.facebook.com/EPI-USE_Labs)