Initial Client Setup Custom

PLUG-IN : ***it.cnet.impl.InitClientSetup\_Custom***

Processo di inizializzazione dei dati a livello Client e/o System.

Classe java: ***it.cnet.idempiere.InitClientSetup\_Custom.process.InitialClientSetup\_Custom***

I dati salvati all’interno di 2Pack creati a livello Client, vengono posizionati in una cartella specifica all’interno del server-idempiere.

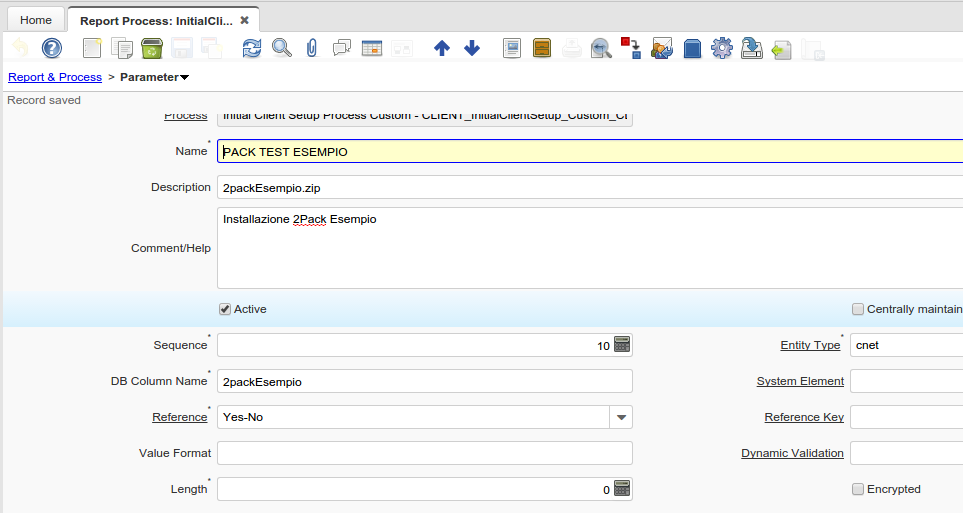
Ogni 2Pack vengono parametrizzati all’interno del processo, con il loro specifico nome; in base ai 2Pack scelti nella maschera del processo, vengono letti all’interno della cartella e caricati sull’application.

A livello System, nella maschera ‘Report & Process’ sono stati creati 2 processi che richiamano la stessa classe java:

* ***Initial Client Setup Process Custom - CLIENT*** = per questo processo settato il campo “Data Access Level(*AccessLevel*) come - Client+Organization - ; questo processo viene chiamato solo per i 2Pack Tenant(o dati a livello Client…)
* ***Initial Client Setup Process Custom - SYSTEM***= per questo processo settato il campo “Data Access Level(*AccessLevel*) come - System only - ; questo processo viene chiamato solo per i 2Pack dati a livello System, come creazioni di nuove tabelle, regole a livello System, etc..

Nella sottomaschera ‘Parameter’ si dovranno specificare i nomi dei file fisici 2Pack presenti nella cartella, che una volta lanciato la maschera di processo si potranno scegliere tramite selezione di checkbox.

NOTA BENE:

**CAMPO “Name(Nome)”** = nome/descrizione che compare a video nella maschera del processo

**CAMPO “Description(descrizione)”** = qui **SI DEVE SPECIFICARE** il nome esatto del path del file 2Pack presente in cartella, preceduto da “file:///….” (specifica che serve per il processo…) ; i file devono essere assolutamente senza spazi o con “\_”

**CAMPO “DB Column Name”** = qui **SI DEVE SPECIFICARE** il nome del file SENZA l’estensione; questo serve per il processo creato per identificare ogni singolo 2Pack…

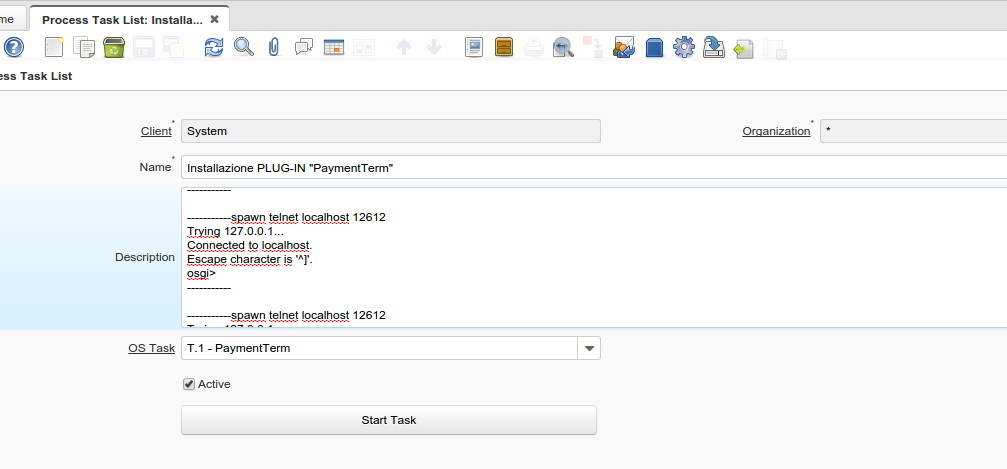
**CAMPO “Reference”** = qui settare il tipo “Yes-No”, per abilitare scelta d’installazione del 2Pack

Il processo in questione, controlla la referenzialità dei file: controlla che tutti i 2Pack scelti a maschera siano giusti come nomi file e che siano presenti nella cartella e POI DOPO lancia il packIn dei file….

# Installazione plug-in via Task.

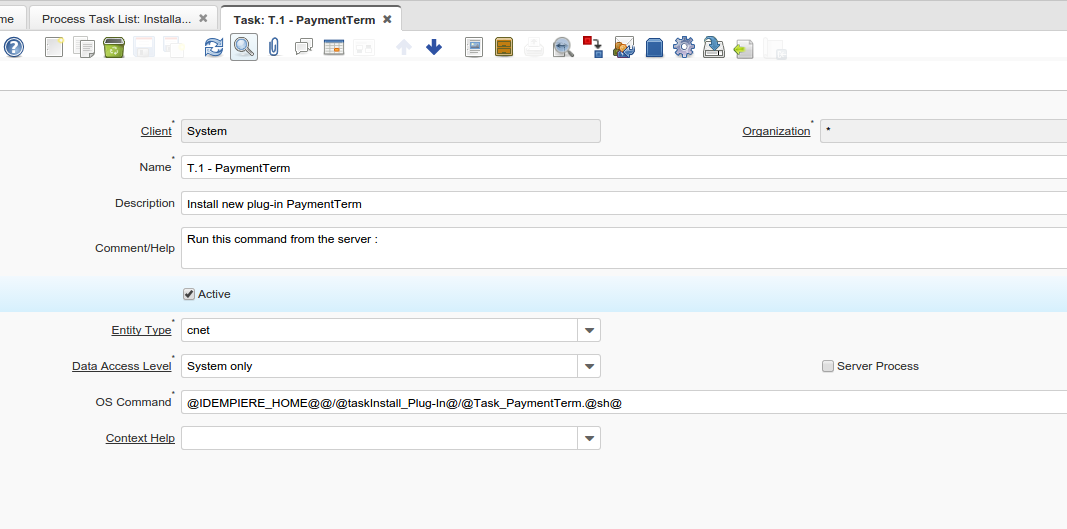
Creazione di una nuova maschera per il lancio dei task di installazione Plug-in

- PROCESS TASK LIST -



Da questa maschera viene attivato il task che si vuole lanciare e ad avvenuta esecuzione viene salvato nella “description” il risultato dell’esecuzione.

Per far funzionare il processo di avvio task, occorre impostare la task corrispondente al plug-in che si vuole installare.



Nel campo “OS Command” specificare il path completo dello script \*.sh.

***È importante specificare ogni script \*.sh per ogni plug-in***; come da esempio, il file /opt/repos/idempiere-download/taskInstall\_Plug-In/Task\_PaymentTerm.sh:

#!/usr/bin/expect -f

/usr/bin/expect << EOF

spawn telnet localhost 12612

expect "$ "

send -- "install file:/opt/repos/idempiere-download/plugin-italia/payment-term/it.cnet.impl.payment\_term.jar\n"

expect "$ "

send -- "start it.cnet.impl.payment\_term\n"

expect "$ "

send -- "disconnect\n"

expect "$ y\n"

EOF

Ricordarsi che il nome del jar, deve essere privo del numero versionamento+data……

**NOTA: Come lanciare gli Script sql via TASK**

1. Rinominare lo script .sql in .sh (ad esempio *tasksystem-ad\_org\_default.sh*)
2. all’interno del file .sh dobbiamo inserire i seguenti comandi

**psql -d idempiere -U adempiere << EOF**

…………….codice sql…………….

**EOF**