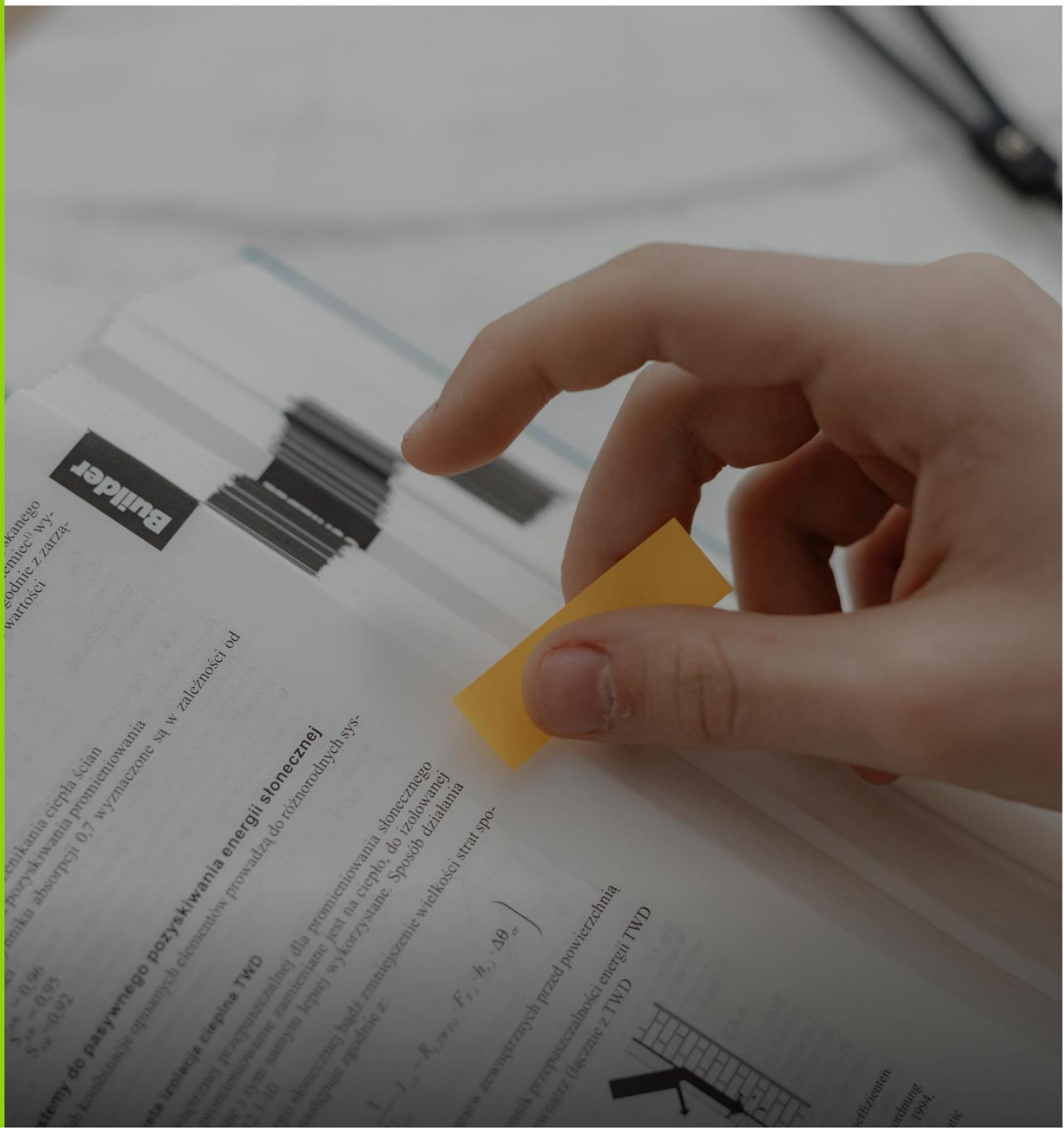
LOOkie Data Bridge

Manuale operativo/*User Manual*



Sommario

Revisioni 5
Introduzione 6
Estensione 6
Requisiti tecnici 6
Dashboard 7
Setup 8
Campi 8
Numeratore Trasmittenti 8
Numeratore Operazioni 8
Setup Utente 8
Ricetrasmettitori 9
Introduzione 9
Campi 10
Strutture 11
Introduzione 11
Campi 11
Proprietà Struttura 12
Introduzione 12
Campi 12
Azioni 13
Operazioni 14
Introduzione 14
Campi 14
Descrizione dettagliata elementi Dashboard 17
Esempi Struttura 18
SERIALIZZAZIONE ARTICOLO 18
DESERIALIZZAZIONE ARTICOLO 20
SERIALIZZAZIONE ORDINI VENDITA 22
FUNZIONI 30
FIXED 30
Parametri Associati 30
FORMAT 30
Parametri Associati 30
TRANSFORMATION 30
Parametri Associati 30

MODULI 31
BASE 31
Comandi 31
REPORT 31
Comandi 31
REST 32
Comandi 32
AUTENTICAZIONE 34
PARAMETRI OBBLIGATORI 34
AUTHORIZATION_TYPE 34
BASIC 34
TRASFORMAZIONE 35
Introduzione 35
Campi 35
API 36
ESEMPIO CHIAMATE API 37
SERIALIZE 37
DESERIALIZE 38
CODA PROCESSI 39
WORKFLOW 41
EVENTO 41
CONDIZIONE 41
RISPOSTA 42
ESEMPIO DI UTILIZZO 43

Revisioni

Data	Autore	Versione
11/07/2023	Marco Costanza	Documento principale
26/07/23	Marco Costanza	Adeguamento versione 1.0.21

Introduzione

Lookie Data Bridge è l'estensione pensata per semplificare l'interazione tra Business Central e i servizi esterni, migliorando allo stesso tempo il sistema di ricezione dati e la successiva elaborazione degli stessi e consentendo l'invio integrato a servizi esterni tramite chiamate rest.

Senza necessità di scrivere codice ad hoc o di creare pagine API personalizzate, tramite un setup può modellarsi sul tracciato di scambio parametrizzando i collegamenti con le tabelle di sistema.

Il presente documento intende illustrare le principali configurazioni necessarie ad un consulente applicativo per rendere operativa nella sua versione base l'estensione.

L'applicazione è pensata per essere una struttura modulare estendibile. L'installazione di questa estensione mette a disposizione alcuni moduli ciascuno dei quali ha dei comandi base che possono essere utilizzati.

La possibilità di integrare moduli aggiuntivi, la cui creazione è molto semplice, estende questa possibilità arricchendo il sistema di nuovi strumenti.

Estensione

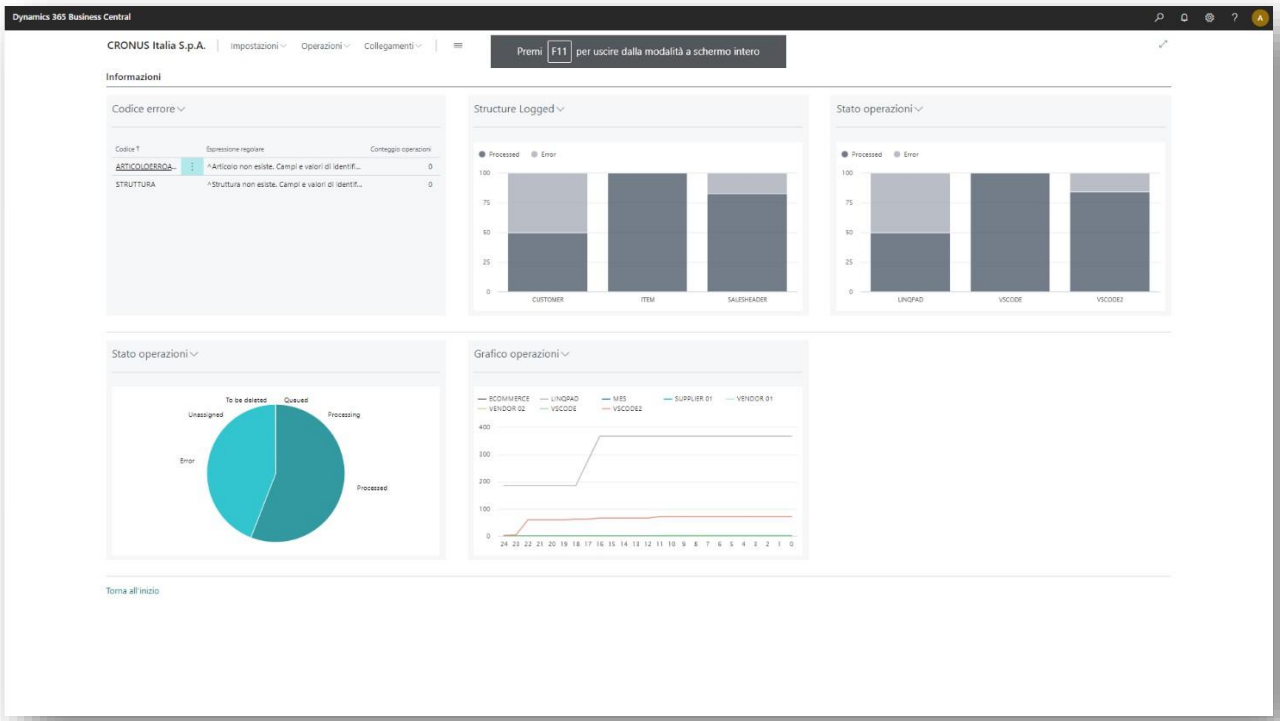
Il modulo base dell'estensione **Lookie Data Bridge** non ha dipendenze e può essere installato su qualsiasi versione di Microsoft Dynamics Business Central 365 a partire dalla versione 21.

Requisiti tecnici

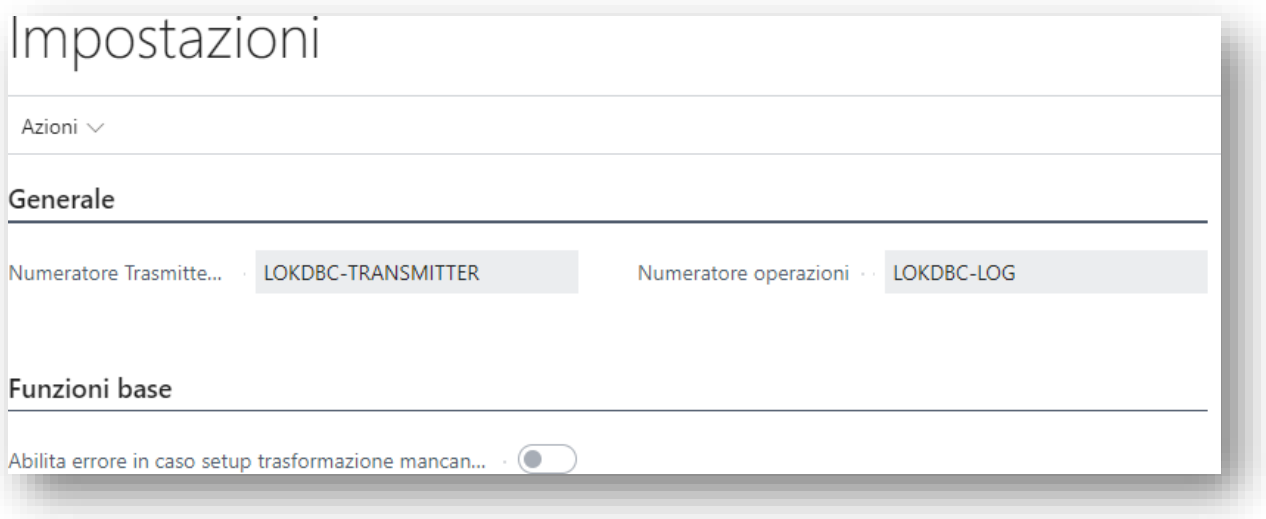
Dashboard

L'utente responsabile della supervisione del processo di scambio dati, da e per il sistema di gestione, avrà a disposizione una dashboard dedicata che sarà associata al suo ruolo utente. Questa dashboard fornirà una visione riassuntiva dello stato dei diversi scambi di dati con i vari interlocutori presentando le informazioni in una forma grafica interattiva dando accesso a una lista aggregata degli errori riscontrati, consentendogli di intervenire prontamente per risolverli e un menù per l'accesso diretto ai vari elementi.

La dashboard rappresenta uno strumento essenziale per il monitoraggio e la gestione efficiente del flusso di dati, consentendo all'utente di avere una panoramica chiara e immediata delle attività in corso e dei potenziali problemi da affrontare.



Setup



Campi

Numeratore Trasmittenti

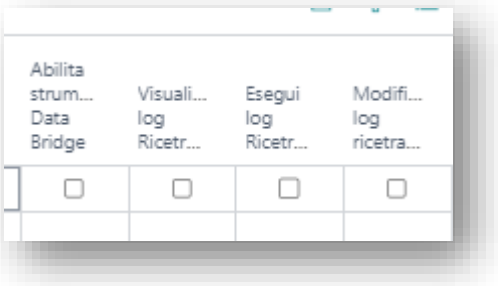
Il numeratore di default per i ricetrasmettitori che verranno creati.

Numeratore Operazioni

Il numeratore che viene assegnato ai singoli ricetrasmettitori generat~~ei~~ per essere usato per gli identificatori univoci operazione.

Setup Utente

Perché un utente possa utilizzare l'applicazione è necessario impostare dei permessi. A livello di setup utente è possibile dare ad un singolo utente accesso a tutte le operazioni a prescindere da quale sia il ricetrasmettitore legato all'operazione.



Si può limitare la visualizzazione, l'esecuzione e la modifica dei record sia per il dato in ingresso che per i parametri associati.

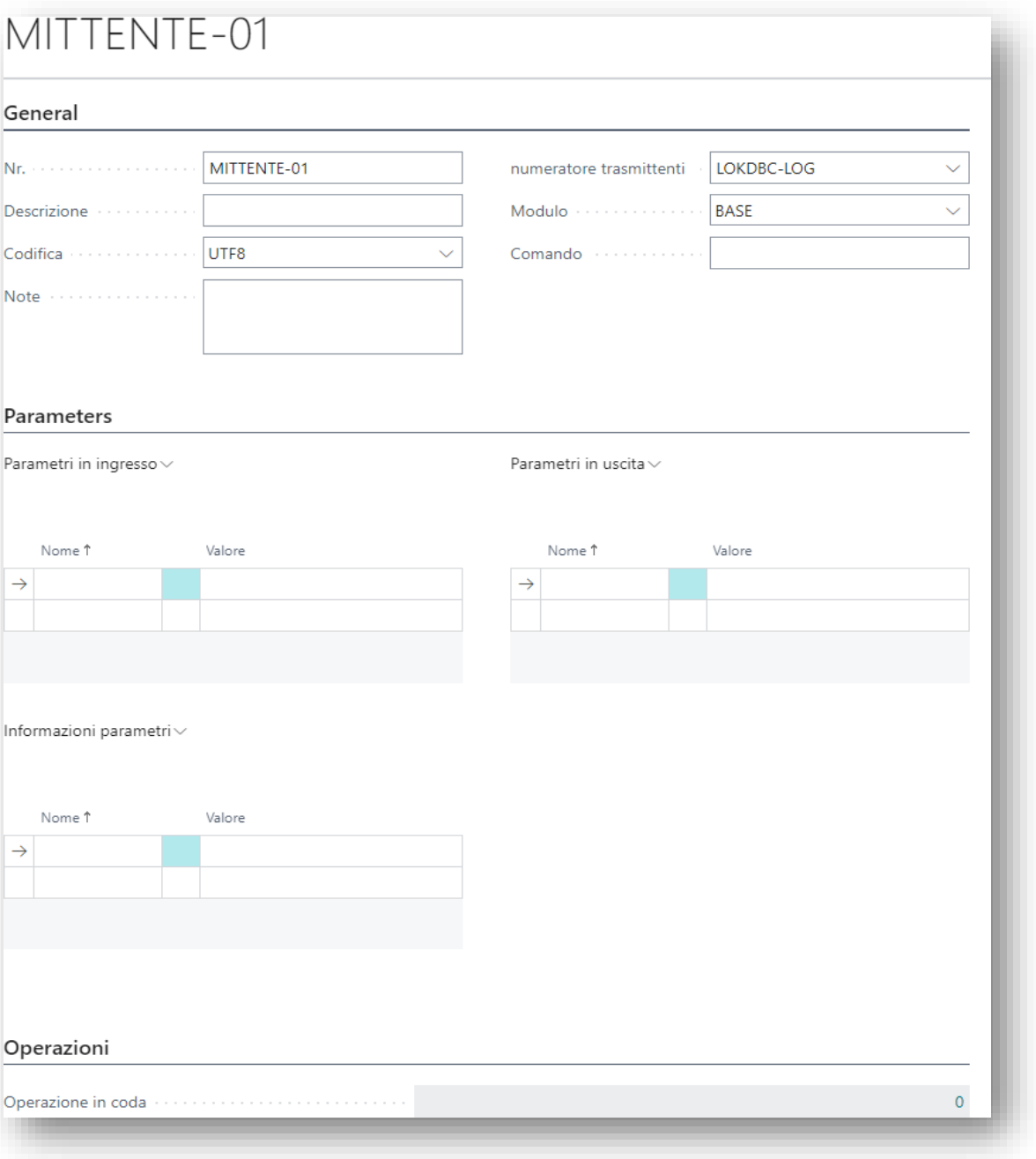
Il flag "Abilita strumenti Data Bridge" attiva funzionalità aggiuntive di controllo, non necessarie per l'utente dell'applicazione ma utili in fase di sviluppo e controllo delle funzionalità.

Qualora non si volesse dare permessi a livello di intera applicazione è possibile limitare le funzionalità per singolo ricetrasmettitore dalla scheda dedicata.

Ricetrasmettitori

Introduzione

Una ricetrasmettitore è uno degli interlocutori in una operazione di scambio dati. Ogni servizio esterno che intenda inviare dati verso Business Central utilizzando le funzionalità messe a disposizione dalla Lookie Data Bridge dovrà essere censito in questa Tabella.



Ad ogni ricetrasmettitore sono associati un numero identificativo univoco, una descrizione e delle note estese.

Campi

Codifica

Il campo codifica definisce l'encoding usato per il dato che deve essere scambiato e in che modo questo verrà salvato e gestito nel corso della sua vita. Il valore di default è UTF8 ma può essere variato in tutte le possibili modalità utilizzabili da business central.

Numeratore

È il numeratore utilizzato dal sistema per generare l'ID univoco delle operazioni relative allo scambio con quest~~a~~o ricetrasmettitore. Alla creazione ad ogni nuovo record viene assegnato lo stesso numeratore ma è possibile impostarne uno personalizzato in modo da avere numerazioni diverse in base ai vari flussi di scambio.

Modulo

Rappresenta il modulo di default associato al~~la~~ ricetrasmettitore. In questo modo, nella ricezione dati, non è necessario fornire questa informazione. Il sistema, in fase di ricezione dati via API, provvederà ad interpretare i dati utilizzando questo modulo. Nella sua versione BASE, c'è solo il modulo BASE. L'aggiunta di moduli aggiuntivi è delegata all'installazione di app dipendenti da questa che ne estendono le funzionalità.

Comando

Definisce il comando di default da associare al~~la~~ ricetrasmettitore tra quelli disponibili nel modulo di default selezionato nel campo "Modulo". Il sistema, in fase di ricezione dati via API, provvederà ad interpretare i dati utilizzando questo comando. Il campo, in questo momento è a inserimento libero e si delega l'applicativo che si occupa della configurazione di conoscere il setup corretto da usare.

Parametri

Ogni operazione ha associato un modulo e un comando. I comandi possono avere dei parametri necessari all'elaborazione dell'operazione. Questi sono normalmente associati alla singola riga operazione. Tuttavia, è possibile impostare nella scheda del ricetrasmettitore i parametri di default che verranno usati su ogni operazione associata a meno che non siano diversamente specificati sulla operazione stessa.

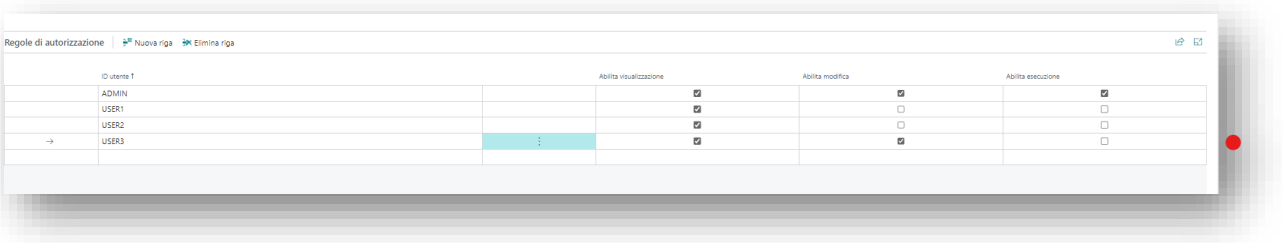
I parametri utilizzabili sono principalmente di tre tipi:

- Ingresso
- Uscita
- Informativi

I parametri di ingresso sono quelli che verranno passati al comando dell'operazione. I parametri in uscita sono il risultato dell'operazione stessa.

I parametri informativi non hanno applicazione pratica ma vanno interpretati come note all'operazione.

Permessi



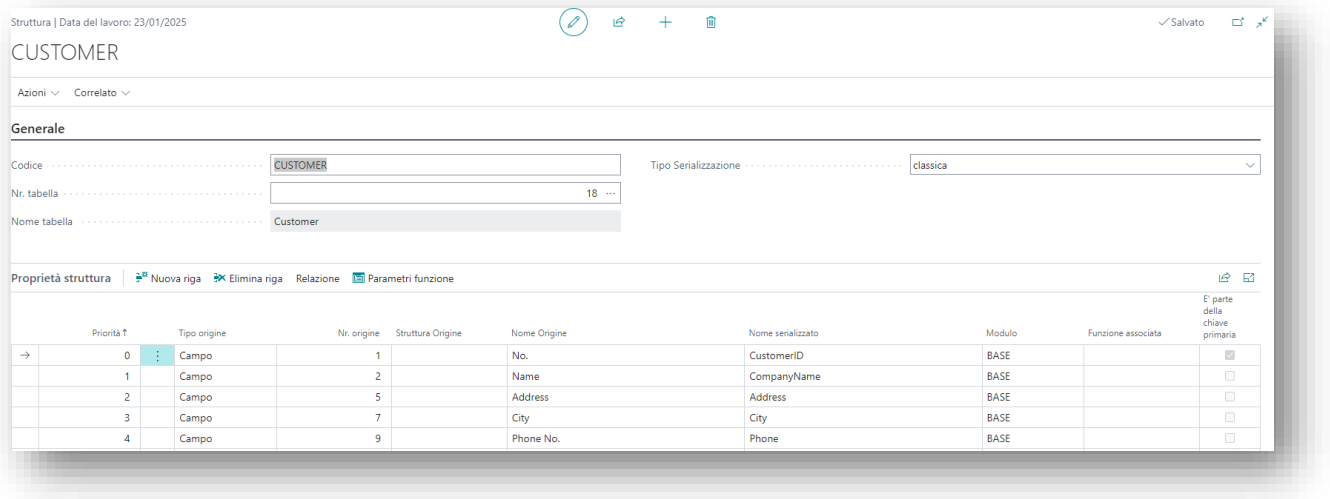
A livello di singolo ricetrasmettitore è possibile impostare i permessi di visualizzazione, esecuzione e modifica dei dati presenti nelle varie operazioni. Questi setup impattano anche a livello di dashboard.

Strutture

Introduzione

Le strutture sono le entità di scambio tra Business Central e il sistema esterno permettono di definire come quali informazioni saranno estratte estrapolate dai dati ricevuti dall'API esposta e come le singole informazioni saranno mappate sulle tabelle e sui campi del gestionale.

Ogni struttura ha un codice identificativo univoco ed è associabile a una sola tabella.



Campi

Codice

Il codice identificativo univoco della struttura.

Nr. Tabella

La tabella business central a cui la struttura è collegata.

Nome Tabella

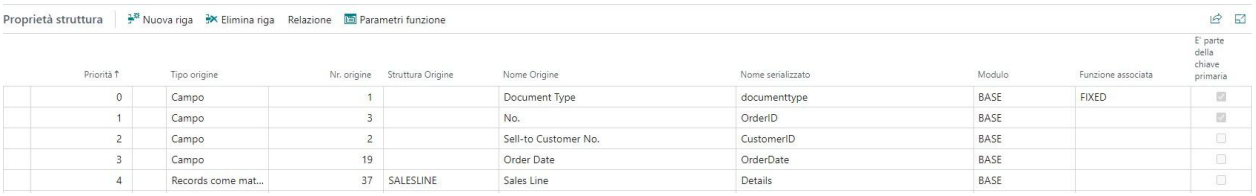
Campo di lookup che indica il nome della tabella selezionata.

Tipo Serializzazione

Definisce il nome con cui saranno serializzate le varie proprietà partendo dal nome del campo o della tabella da gestire. È una impostazione di default e ovviamente ogni singolo elemento può essere gestito manualmente indipendentemente.

- **Classica**: i nomi sono minificati (ES: Sell-to Customer No. → selltocustomerno)
- **NAV**: i nomi seguono la nomenclatura classica di NAV (ES: Sell-to Customer No. →
- Sell_to_Customer_No_)

Proprietà Struttura


Proprietà struttura | Nuova riga | Elimina riga | Relazione | Parametri funzione

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato	Modulo	Funzione associata	E' parte della chiave primaria
0	Campo	1		Document Type	documenttype	BASE	FIXED	☑
1	Campo	3		No.	OrderID	BASE		☑
2	Campo	2		Sell-to Customer No.	CustomerID	BASE		☐
3	Campo	19		Order Date	OrderDate	BASE		☐
4	Records come mat...	37	SALESLINE	Sales Line	Details	BASE		☐

Introduzione

Ad ogni struttura sono associate n righe proprietà ciascuna delle quali rappresenta un elemento di dettagli ~~della struttura sui~~o della struttura su cui si sta lavorando. Può trattarsi di un campo della tabella stessa o di una ulteriore tabella collegata alla principale. In questo modo è possibile creare strutture multilivello che descrivano in maniera completa un oggetto.

Campi

Priorità

Definisce l'ordine di processamento di ogni singol~~o~~a proprietà. In fase di serializzazione definisce l'ordine in cui le proprietà saranno presenti nel tracciato. In fase di deserializzazione definisc~~ono~~e l'ordine con cui i campi vengono validati in fase di importazione. Questa sequenza è rilevante perché la validazione di un campo spesso ne valorizza altri quindi, volendo so~~prappo~~vrappor re eventuali valori inseriti via codice da validazioni precedenti è opportuno che la proprietà da cui si vuole estrarre il dato venga elaborata successivamente.

Tipo Origine

Rappresenta la tipologia di dato della struttura che la riga andrà ad elaborare.

- **Campo:** un campo della tabella associata (ES: "Sell-to Customer No" nella tabella "Sales Header")
- **Records come matrice**: una tabella collegata con un numero multiplo di record al suo interno collegati al~~la~~ record in gestione nella struttura padre (ES: "Sales Line" con la tabella "Sales Header")
- **Record singolo come oggetto:** una tabella collegata al cui interno c'è un solo record collegato al record in gestione nella struttura padre (ES: "Payment Method" con la tabella "Sales Header")
- **Singola Proprietà:** una tabella collegata al cui interno c'è un solo record collegato al record in gestione nella struttura padre (ES: "Payment Method" con la tabella "Sales Header") ma di questa tabella solo una proprietà viene recuperata. È quello che in business central viene effettuato con un campo di tipo lookup

Nr. Origine

Definisce l'oggetto da cui recuperare l'informazione necessaria~~. I~~, in base alla tipologia di origine specificata. Può essere il numero di un campo o il numero di una tabella.

Struttura Origine

Valido solo per le righe di tipo Record singolo come oggetto, Records come matrice e Singola Proprietà e definisce la struttura da utilizzare per recuperare i dati dal record associato.

Nome Origine

Nome del campo o della tabella impostati nel campo nr. origine.

Nome Serializzato

Il nome dell'oggetto serializzato collegato all'oggetto selezionato nel tipo origine.

Modulo

Definisce se sulla proprietà della riga deve essere aggiornata una funzione di manipolazione del dato e da quale modulo questa deve essere recuperata.

Funzione Associata

La funzione di manipolazione del dato associata alla riga. Vedi sezione FUNZIONI per le informazioni relative.

Parte Della Chiave Primaria

Campo calcolato in tempo reale che definisce se la riga in questione è relativa a un campo e se questo è parte della chiave primaria della tabella associata alla struttura.

Azioni

Serializza - Deserializza

Permette di selezionare un record dalla tabella impostata nel campo nr. Tabella per ottenere il dato serializzato secondo le specifiche impostate nelle proprietà della struttura. Vedi sezione esempi per ulteriori dettagli.

Relazioni

Permette di definire i legami tra una riga di proprietà della struttura associata a una tabella accessoria e il record in gestione nella struttura padre.



- **Valore collegato:** è il campo della tabella collegata che si vuole usare per creare una relazione con la tabella padre.
- **Tipo relazione:** è il tipo di gestione che si vuole usare per gestire il campo nella colonna valore collegato può valere CONST, FILTER o FIELD.
- **Valore relazione:** se il tipo relazione è FIELD rappresenta il campo della tabella padre con cui lo si vuole collegare. Se il tipo è CONST rappresenta il valore costante da usare nell'individuare i record della tabella collegata. Se il tipo è FILTER viene usato il formato filtri standard di Business central (ES: I*).

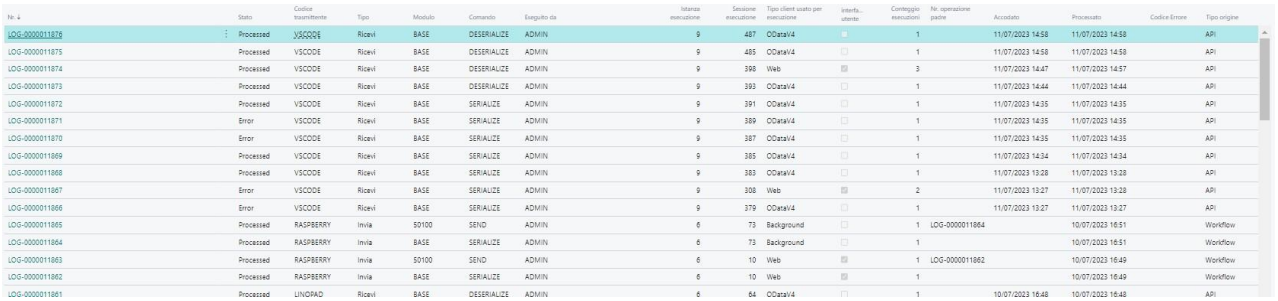
Nell'esempio la tabella Sales Line è collegata alla tabella Sales Header dai campi "Document Type" e dal campo "Document No". Sulle righe che corrisponde al campo "No." sulla tabella padre.

Operazioni

Introduzione

Ogni singola comunicazione inviata tramite API al sistema viene loggata nella tabella delle operazioni. Un'operazione rappresenta una operazione atomica di gestione del dato in base alla parametrizzazione data dal codice "Ricetrasmettitore", dal modulo da applicare e dal comando scelto all'interno di quelli disponibili per il modulo.

Ogni operazione può avere una serie di parametri di input aggiuntivi ad uso del comando scelto.



Nr.	Stato	Codice trasmittente	Tipo	Modulo	Comando	Eseguito da	Istanza esecuzione	Sessione esecuzione	Tipo client usato per esecuzione	Interfa... utente	Conteggio esecuzioni	Nr. operazione padre	Accodato	Processato	Codice Errore	Tipo origine
LOG-0000011876	Processed	VSCODE	Ricevi	BASE	DESERIALIZE	ADMIN	9	487	ODataV4		1		11/07/2023 14:58	11/07/2023 14:58		API
LOG-0000011875	Processed	VSCODE	Ricevi	BASE	DESERIALIZE	ADMIN	9	485	ODataV4		1		11/07/2023 14:58	11/07/2023 14:58		API
LOG-0000011874	Processed	VSCODE	Ricevi	BASE	DESERIALIZE	ADMIN	9	398	Web		3		11/07/2023 14:47	11/07/2023 14:57		API
LOG-0000011873	Processed	VSCODE	Ricevi	BASE	DESERIALIZE	ADMIN	9	393	ODataV4		1		11/07/2023 14:44	11/07/2023 14:44		API
LOG-0000011872	Processed	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	391	ODataV4		1		11/07/2023 14:35	11/07/2023 14:35		API
LOG-0000011871	Error	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	389	ODataV4		1		11/07/2023 14:35	11/07/2023 14:35		API
LOG-0000011870	Error	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	387	ODataV4		1		11/07/2023 14:35	11/07/2023 14:35		API
LOG-0000011869	Processed	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	385	ODataV4		1		11/07/2023 14:34	11/07/2023 14:34		API
LOG-0000011868	Processed	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	383	ODataV4		1		11/07/2023 13:28	11/07/2023 13:28		API
LOG-0000011867	Error	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	308	Web		2		11/07/2023 13:27	11/07/2023 13:28		API
LOG-0000011866	Error	VSCODE	Ricevi	BASE	SERIALIZE	ADMIN	9	379	ODataV4		1		11/07/2023 13:27	11/07/2023 13:27		API
LOG-0000011865	Processed	RASPBERRY	Invia	50100	SEND	ADMIN	6	73	Background		1	LOG-0000011864		10/07/2023 16:51		Workflow
LOG-0000011864	Processed	RASPBERRY	Invia	BASE	SERIALIZE	ADMIN	6	73	Background		1			10/07/2023 16:51		Workflow
LOG-0000011863	Processed	RASPBERRY	Invia	50100	SEND	ADMIN	6	10	Web		1	LOG-0000011862		10/07/2023 16:49		Workflow
LOG-0000011862	Processed	RASPBERRY	Invia	BASE	SERIALIZE	ADMIN	6	10	Web		1			10/07/2023 16:49		Workflow
LOG-0000011861	Processed	LINOPAD	Ricevi	BASE	DESERIALIZE	ADMIN	6	64	ODataV4		1		10/07/2023 16:48	10/07/2023 16:48		API

Campi

Nr.

Numeratore univoco che definisce la transazione.

Stato

Lo stato della riga. Quando viene inserita è in stato "Queued" e dopo l'esecuzione passa in stato Processato o Errore. Esistono stati intermedi da usare per eventualità specifiche.

Codice ricetrasmettitore

Il codice ricetrasmettitore collegato all'operazione in corso.

Tipo

- Ricevi
- Invia
- Informazione
- Dettagliato

Modulo

Il modulo da associare all'operazione corrente.

Comando

Il comando associato all'operazione corrente tra quelli disponibili all'interno del modulo scelto. Questo permette di avere lo stesso codice identificativo di un comando all'interno di moduli diversi lasciando maggiore libertà di estensione.

Eseguito da

Utente che ha eseguito l'operazione.

Istanza esecuzione

L'istanza del servizio su cui l'operazione è stata eseguita.

Sessione esecuzione

La sessione in cui è stata eseguita l'operazione.

Tipo client usato per esecuzione

Il tipo di client che ha eseguito l'operazione.

- Web: da interfaccia utente
- ODataV4: in fase di ricezione
- Background: da coda processi

Eseguito con interfaccia utente

Definisce se l'esecuzione è avvenuta da interfaccia grafica.

Conteggio esecuzioni

Essendo le varie operazioni modificabili e eseguibili più volte manualmente, questo campo tiene traccia del numero di volte che è stato eseguito il comando sulla riga corrente.

Nr. operazione padre

Se una operazione è stata creata come risultato dell'elaborazione di un'altra operazione in questo campo è indicato il numero di operazione generante.

Accodato

DateTime in cui la riga è stata inserita nella lista operazioni.

Processato

DateTime in cui la riga è stata processata l'ultima volta che è stata eseguita.

Codice Errore

Il codice di errore associato ad un eventuale messaggio ~~l'~~di errore secondo le regular expression presenti nella tabella "Codici Errore".

Tipo origine

Definisce come la riga è stata generata: se è stata una attività dell'utente o un workflow o la co~~s~~da processi ad inserire nel sistema la nuova riga.

Messaggi in ingresso

Un eventuale dato in ingresso su cui effettuare l'operazione~~.~~, in arrivo dal campo input dell'API. Visibile nella factbox della lista o all'interno della scheda operazione.

Messaggi in uscita

Un eventuale risultato dell'operazione effettuat~~e~~a restituita all'API nella proprietà output. Visibile nella factbox della lista o all'interno della scheda operazione.

Parametri in ingresso

Eventuali parametri in ingresso necessari al command per eseguire l'operazione. Visibile nella factbox della lista o all'interno della scheda operazione.

Parametri in uscita

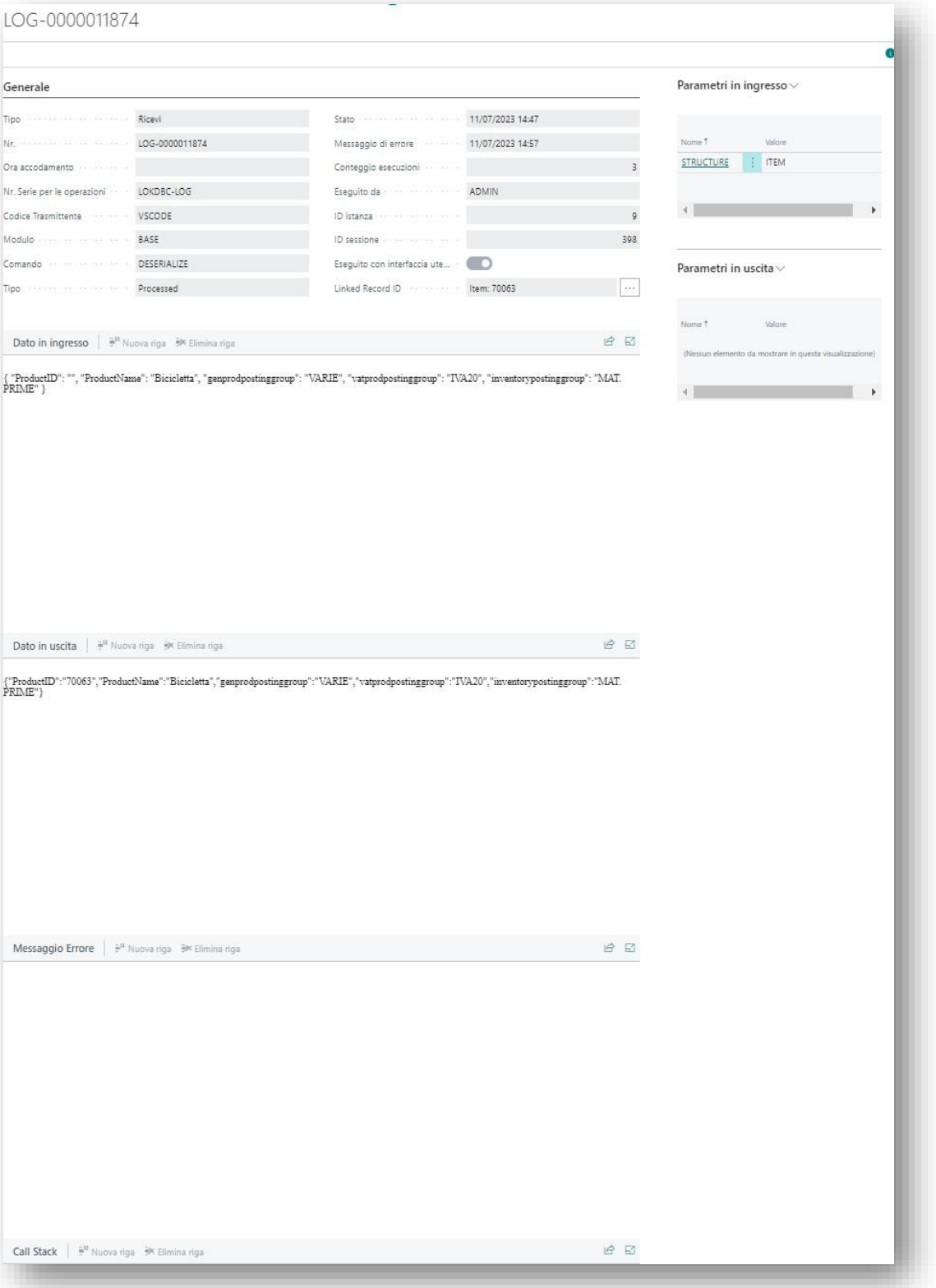
Eventuali parametri risultanti dall'operazione. Visibile nella factbox della lista o all'interno della scheda operazione.

Messaggi in errore

Messaggio di errore qualora ci sia un errore nell'esecuzione della transazione. Visibile nella factbox della lista o all'interno della scheda operazione.

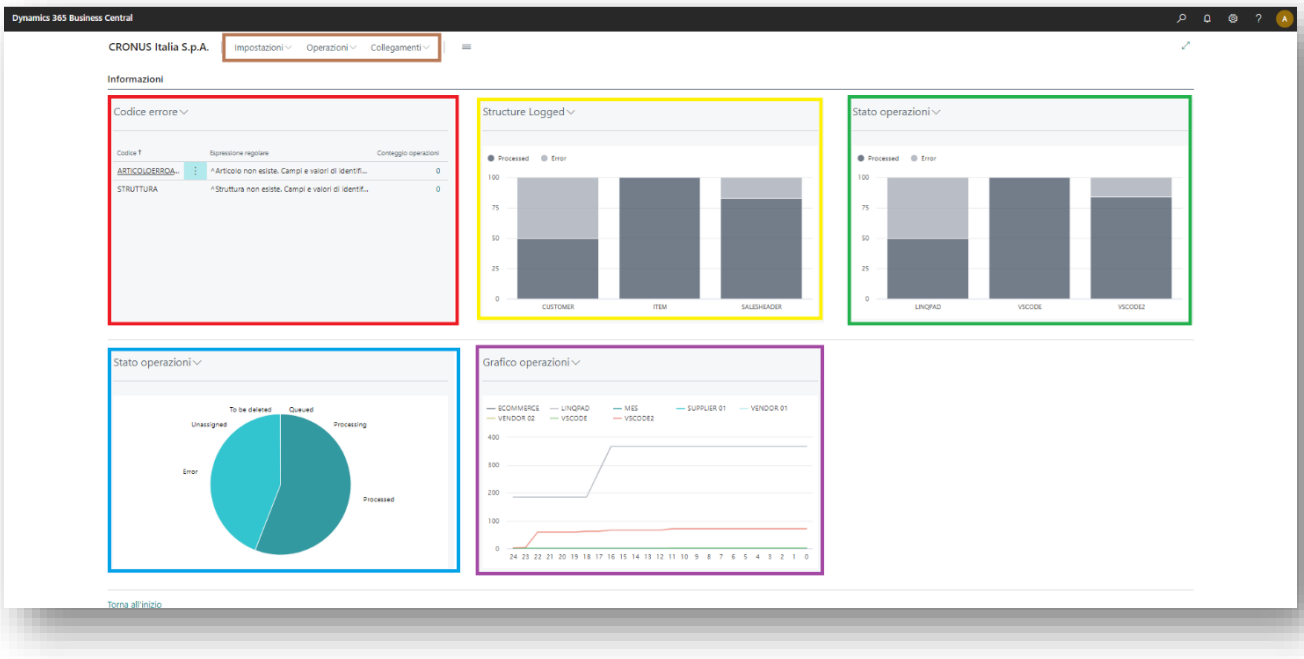
Callstack

Callstack del messaggio di errore qualora ci sia un errore nell'esecuzione della transazione. Visibile nella factbox della lista o all'interno della scheda operazione.



Descrizione dettagliata elementi Dashboard

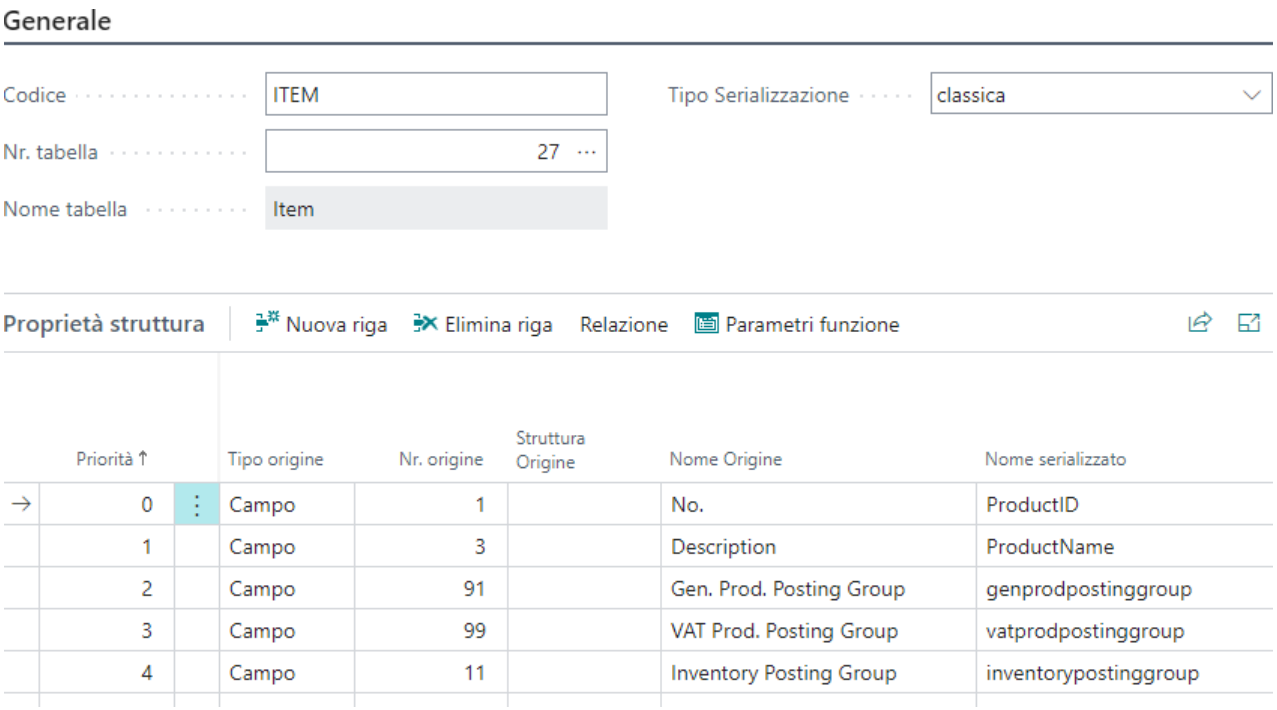
Nella sua configurazione di default al netto delle personalizzazioni la dashboard presenta i seguenti elementi.



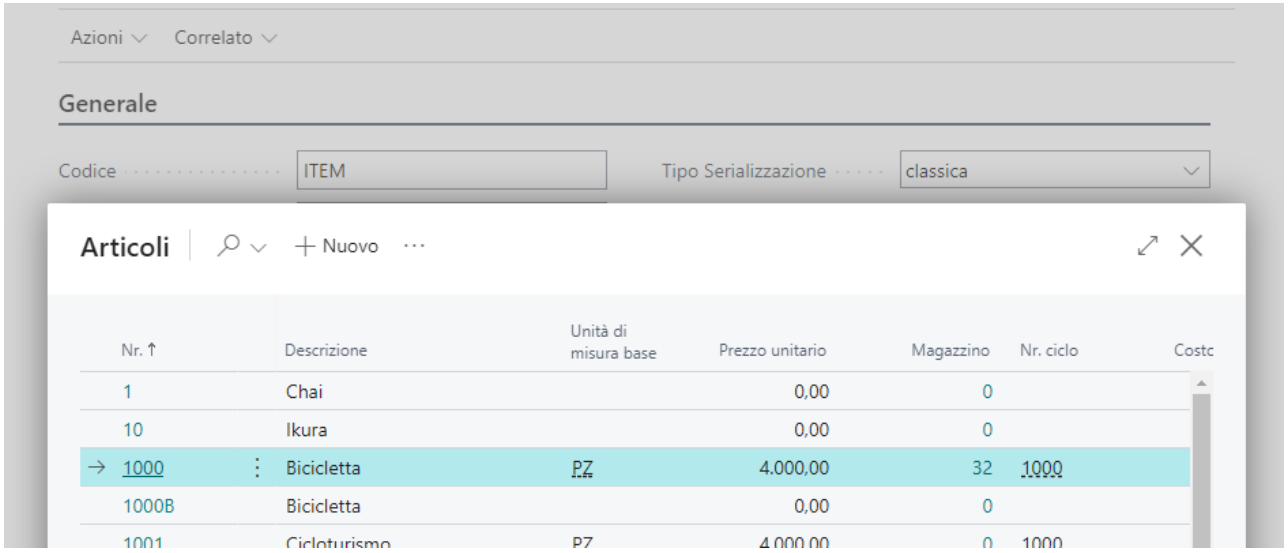
- Lista codici di errori associati alle operazioni presenti nella tabella operazioni.
- Grafico delle operazioni raggruppate per stato e per struttura collegata.
- Grafico delle operazioni raggruppate per stato codice ricetrasmettitore.
- Grafico delle operazioni raggruppato per stato.
- Grafico delle operazioni raggruppato per codice ricetrasmettitore con il numero di operazioni presentei nelle ultime 24 ore.
- Menu di accesso alle varie pagine.

Esempi Struttura

SERIALIZZAZIONE ARTICOLO



Selezionando un record dalla tabella articoli.



Genererà il seguente **json**

```
{
  "ProductID":                                    "1000",
  "ProductName":                                  "Bicicletta",
  "genprodpostinggroup":                          "VARIE",
  "vatprodpostinggroup":                          "IVA20",
```

"inventorypostinggroup": "MAT. ITALIANO PRIME"

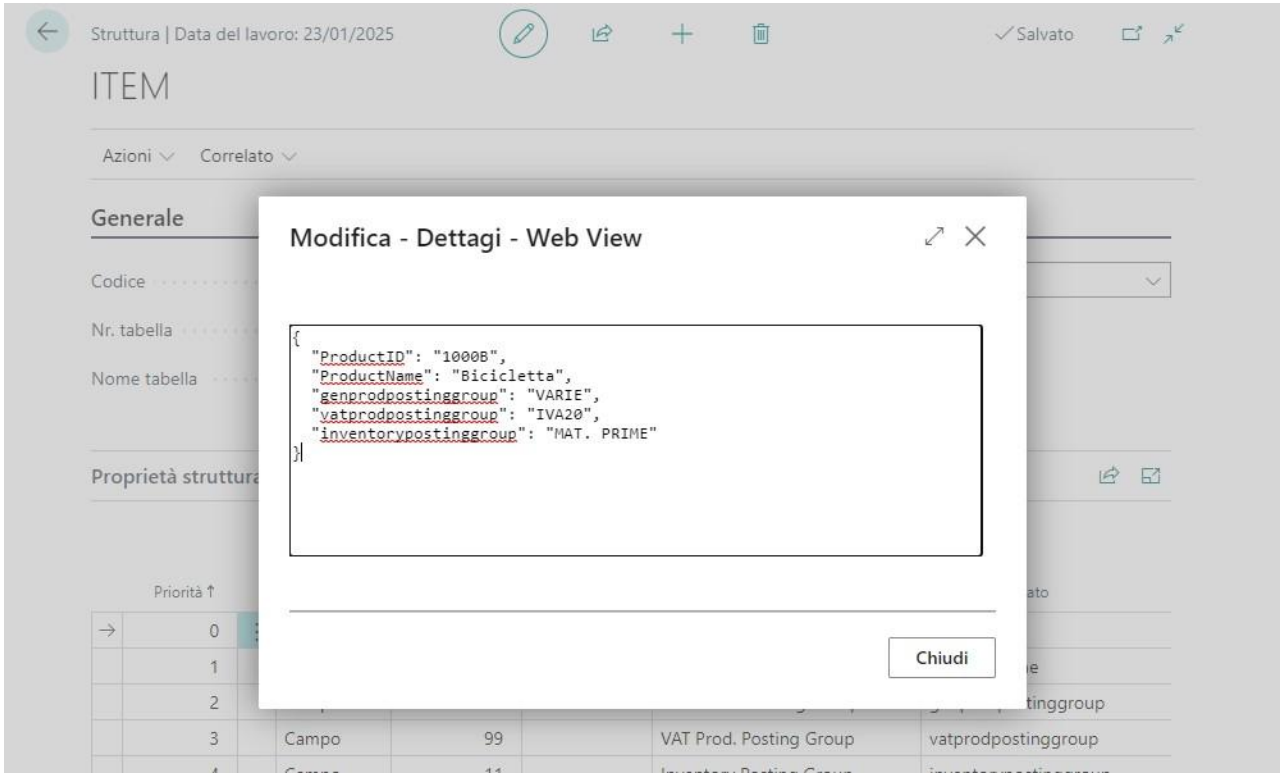
DESERIALIZZAZIONE ARTICOLO

Permette di generare un record nella tabella impostata nel campo “nr. Tabella”, interpretando i dati come da configurazione nelle proprietà.

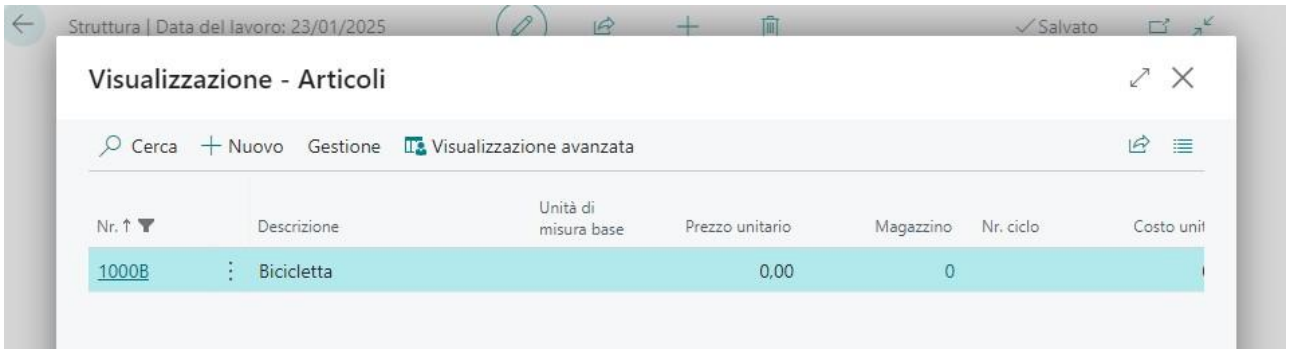
Ad esempio, fornendo in ingresso questo **json.**

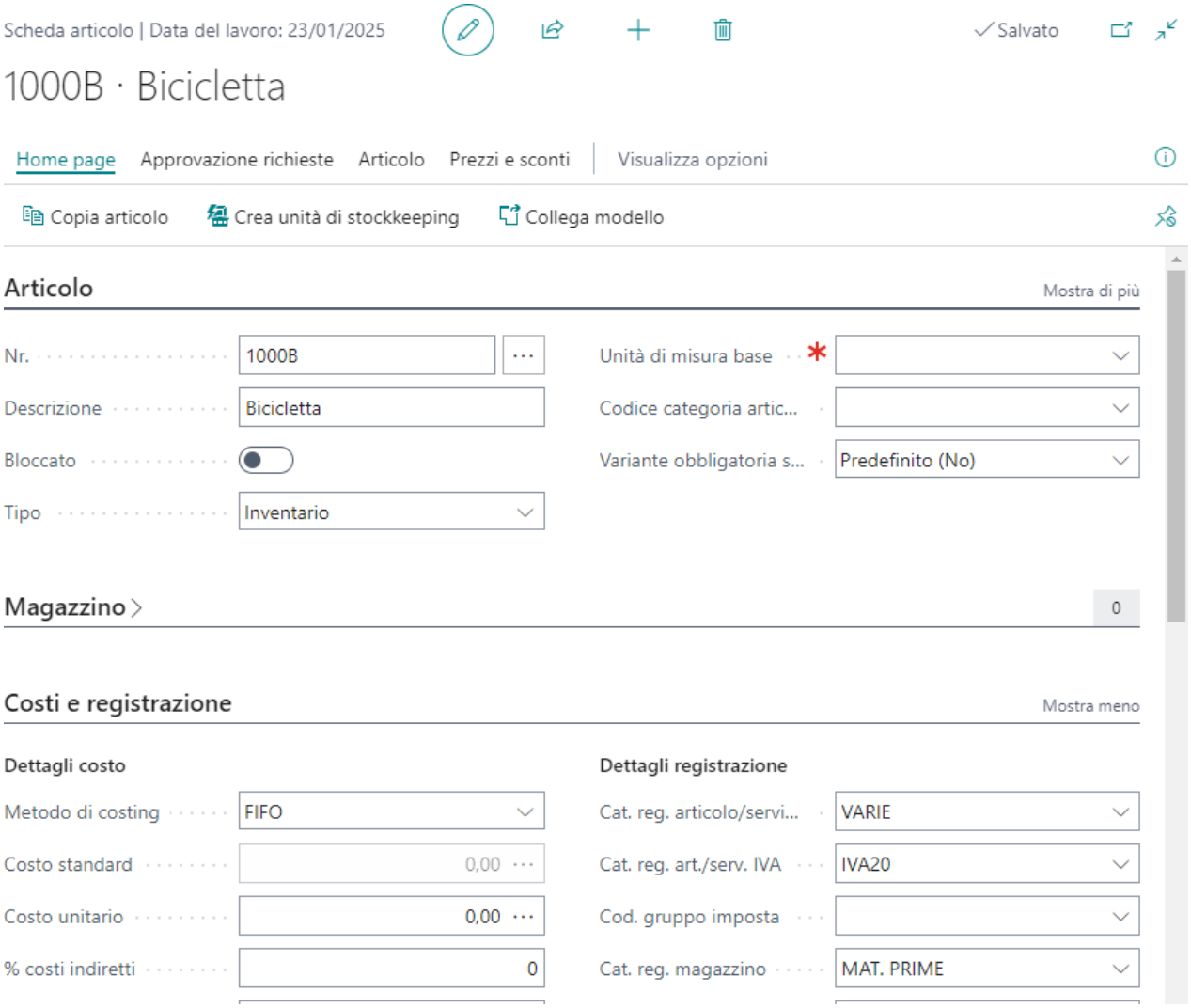
```
{
  "ProductID": "1000B",
  "ProductName": "Bicicletta",
  "genprodpostinggroup": "VARIE",
  "vatprodpostinggroup": "IVA20",
  "inventorypostinggroup": "MAT. PRIME"
}
```

In ingresso



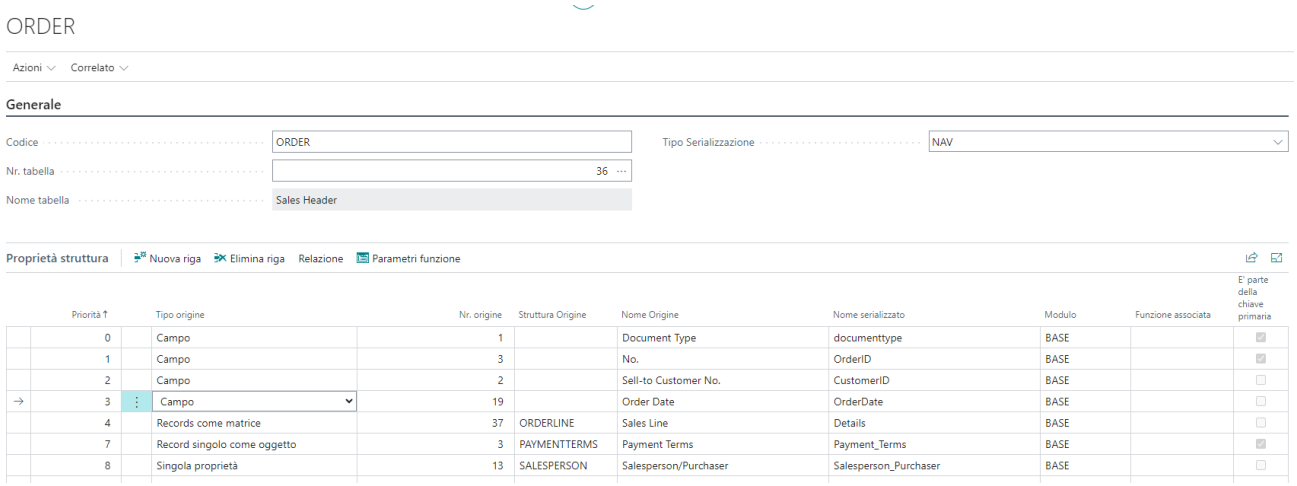
Il sistema genererà il record e aprirà la pagina di lookup associata allo stesso per lasciarlo selezionare.


Struttura | Data del lavoro: 23/01/2025
Salvato
Visualizzazione - Articoli
Cerca
Nuovo
Gestione
Visualizzazione avanzata
Nr.
Descrizione
Unità di misura base
Prezzo unitario
Magazzino
Nr. ciclo
Costo unit
1000B
Bicicletta
0,00
0


Scheda articolo | Data del lavoro: 23/01/2025
Salvato
1000B · Bicicletta
Home page
Approvazione richieste
Articolo
Prezzi e sconti
Visualizza opzioni
Copia articolo
Crea unità di stockkeeping
Collega modello
Articolo
Mostra di più
Nr.
1000B
Descrizione
Bicicletta
Bloccato
Tipo
Inventario
Unità di misura base
Codice categoria artic...
Variante obbligatoria s...
Predefinito (No)
Magazzino
0
Costi e registrazione
Mostra meno
Dettagli costo
Metodo di costing
FIFO
Costo standard
0,00
Costo unitario
0,00
% costi indiretti
0
Dettagli registrazione
Cat. reg. articolo/servi...
VARIE
Cat. reg. art./serv. IVA
IVA20
Cod. gruppo imposta
Cat. reg. magazzino
MAT. PRIME

SERIALIZZAZIONE ORDINI VENDITA

Usiamo un ordine di vendita come esempio per la serializzazione di un singolo documento comprensivo di righe e di dettagli recuperati dai termini di pagamento e il nome dell'agente associato.



La tabella collegata è la 36 Sales Header, il tipo di serializzazione è NAV ma poi alcuni nomi delle proprietà serializzate sono stati modificati.

Anche le righe di dettaglio sono campi recuperati dalla tabella stessa.

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato
0	Campo	1		Document Type	documenttype
1	Campo	3		No.	OrderID
2	Campo	2		Sell-to Customer No.	CustomerID
3	Campo	19		Order Date	OrderDate

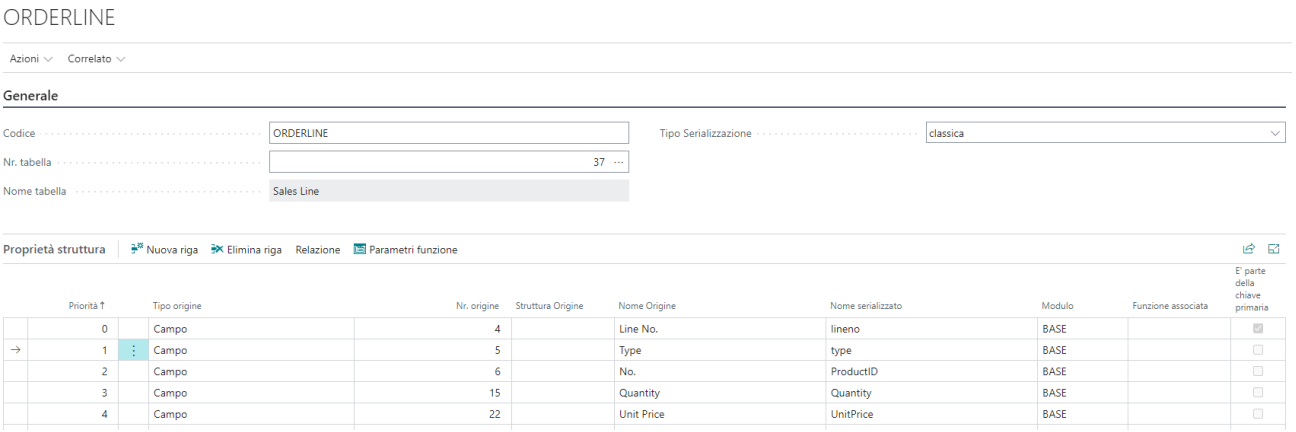
Volendo aggiungere alla struttura anche la parte delle righe vendita è possibile definire una proprietà della struttura come segue.

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato
4	Records come matrice	37		Sales Line	Details

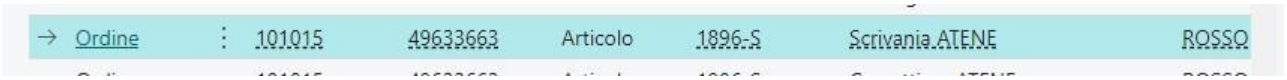
È necessario per prima cosa definire come i dettagli della tabella 37 saranno estrapolati. Quindi va definita una struttura collegata alla tabella 37 da usare per effettuare il mapping.

Il tipo origine è "Records come matrice" perché implica che ci saranno n. record da recuperare e li si vorranno esporre come una matrice di oggetti.

Creando la struttura “Orderline” la si associa alla tabella 37 e si aggiungono le proprietà associate ai campi che si intendono recuperare.



L’operazione di serializzazione, come abbiamo visto nel precedente esempio sugli articoli, scegliendo una riga documento di vendita, genererà il seguente tracciato.




```
{
  "lineno": "30000",
  "type": "2",
  "ProductID": "1896-S",
  "Quantity": "12",
  "UnitPrice": "1005.8"
}
```


A questo punto tornando nella struttura “Order” nella riga associata alla tabella 37 è possibile selezionare il valore della nuova struttura creata.

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato
4	Records come matrice	37	ORDERLINE	Sales Line	Details

Non rimane che impostare le relazioni tra la tabella 37 e la 36 come già visto nella sezione relativa alle relazioni.



A questo punto la serializzazione di un ordine produrrà il seguente output.

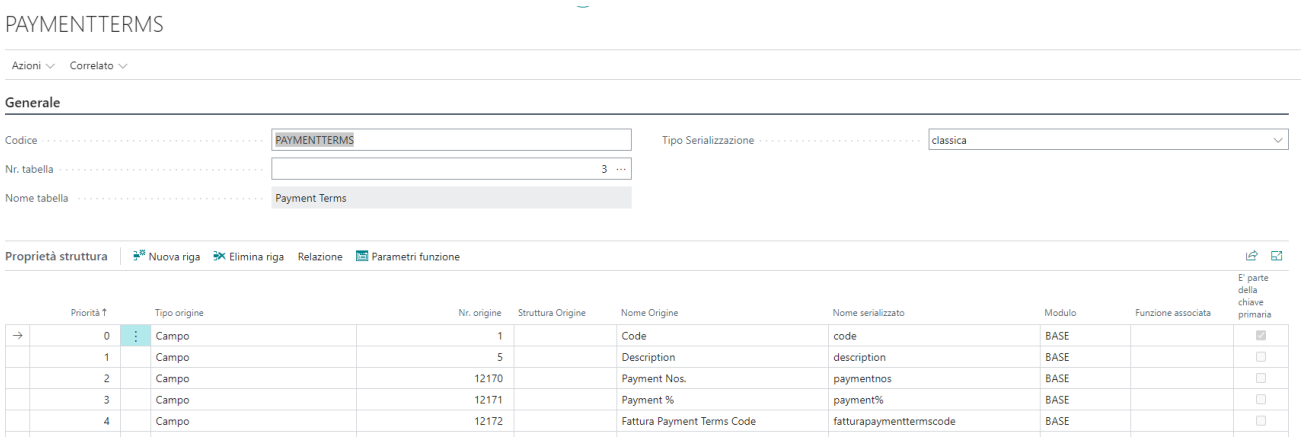

```
{
  "documenttype": "1",
  "OrderID": "10333",
  "CustomerID": "WARTH", "OrderDate":
  "1996-10-18",
  "Sales_Line": [
   {
    "lineno": "2",
    "type": "2",
    "ProductID": "14",
    "Quantity": "10",
    "UnitPrice": "18.6"
   },
   {
    "lineno": "3",
    "type": "2",
    "ProductID": "21",
    "Quantity": "10",
    "UnitPrice": "8"
   },
   {
    "lineno": "4",
    "type": "2",
    "ProductID": "71",
```

```
      "Quantity": "40",
      "UnitPrice": "17.2"
    }
  ]
}
```

Una testata ordine e le righe collegate.

Aggiungiamo una riga legata ai termini di pagamento.

Dopo aver creata una struttura associata alla tabella 3 “Payment terms”.



Possiamo aggiungere una nuova proprietà alla struttura “Order” che faccia riferimento ai termini di pagamento.

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato
7	Record singolo come oggetto	3	PAYMENTTERMS	Payment Term-s	Payment_Terms

E impostare la relazione tra le due.



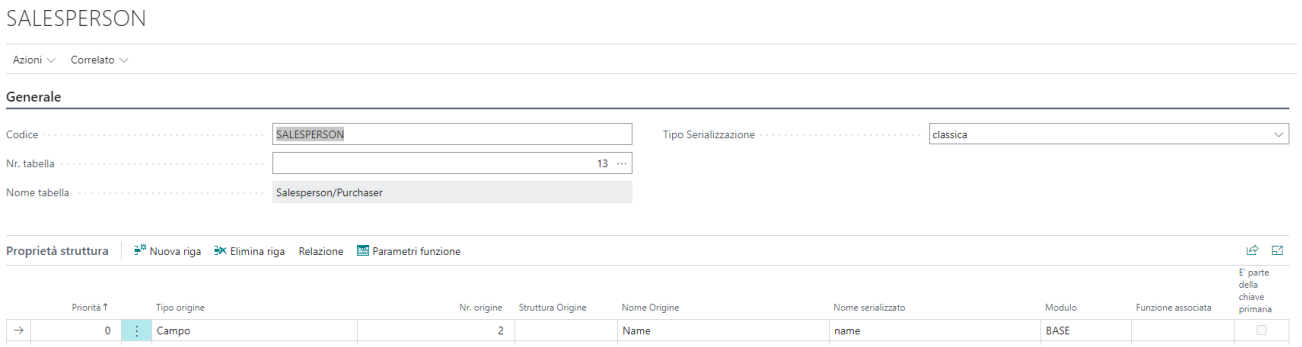
A questo punto la nuova serializzazione produrrà un output che conterrà un singolo oggetto json come valore della proprietà.


```
{
  "documenttype": "1",
  "OrderID": "10333",
  "CustomerID": "WARTH", "OrderDate":
  "1996-10-18",
  "Sales_Line": [
    {
      "lineno": "2",
      "type": "2",
      "ProductID": "14",
      "Quantity": "10",
      "UnitPrice": "18.6"
    },
    {
      "lineno": "3",
      "type": "2",
      "ProductID": "21",
      "Quantity": "10",
      "UnitPrice": "8"
    },
    {
```

```
      "lineno": "4",
      "type": "2",
      "ProductID": "71",
      "Quantity": "40",
      "UnitPrice": "17.2"
    }
  ],
  "Payment_Terms": { "code":
    "21 GIORNI",
    "description": "21 GG data fattura",
    "paymentnos": "1",
    "payment%": "100",
    "fatturapaymenttermscode": ""
  }
}
```


Aggiungiamo una ulteriore riga per recuperare il nome dell'agente associato al documento.

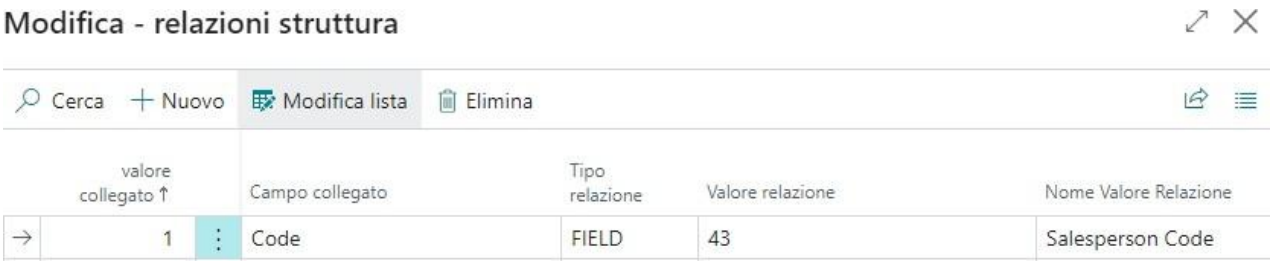
Creata una STRUTTURA associata alla tabella 13 con una sola proprietà, l'unica che sarà recuperata.



Andrà associata all'ordine.

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato
8	Singola proprietà	13	SALESPERSON	Salesperson/Purchaser	Salesperson_Purchaser

È relazionata come già avvenuto nei casi precedenti.




```
{
 "documenttype": "1",
 "OrderID": "10333",
 "CustomerID": "WARTH", "OrderDate":
 "1996-10-18",
 "Sales_Line": [
  {
   "lineno": "2",
   "type": "2",
   "ProductID": "14",
   "Quantity": "10",
   "UnitPrice": "18.6"
  },
  {
   "lineno": "3",
   "type": "2",
   "ProductID": "21",
   "Quantity": "10",
   "UnitPrice": "8"
  },
  {
   "lineno": "4",
   "type": "2",
   "ProductID": "71",
   "Quantity": "40",
   "UnitPrice": "17.2"
  }
 ],
```

```
  "Payment_Terms": { "code":
    "21 GIORNI",
    "description": "21 GG data fattura",
    "paymentnos": "1",
    "payment%": "100",
    "fatturapaymenttermscode": ""
  },
  "Salesperson_Purchaser": "Francesca Milanesi"
}
```


Il json conterrà l'intera struttura dell'ordine.

FUNZIONI

Le funzioni sono esposte dai moduli per permettere elaborazioni aggiuntive dei dati in fase di serializzazione e deserializzazione.

FIXED

Imposta un valore costante da esportare in fase di serializzazione per la proprietà associata alla proprietà struttura. In caso di deserializzazione supplisce alla mancanza di una proprietà o ne sovrascrive il valore.

Parametri Associati

VALUE: il valore costante da usare.

FORMAT

In fase di serializzazione trasforma il dato come da specifiche standard del comando FORMAT di business central come in https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-format-property

Parametri Associati

FORMATPROPERTY: il formato da usare per trasformare il dato (ES: <Year4>-<Month,2>-

<Day,2>)

TRANSFORMATION

Permette di manipolare in ingresso e in uscita il dato in caso di necessità di trascodifica. In ~~p~~fase di configurazione viene definito un modello di traduzione che è quindi applicabile a più campi o tabelle.

Vedi sezione Trasformazione per ulteriori dettagli.

Parametri Associati

MODEL: il modello di trasformazione dati da utilizzare in fase di serializzazione e deserializzazione.

MODULI

In questa versione dell'app sono previsti i seguenti moduli. Ogni modulo rappresenta un insieme di funzionalità messe a disposizione dell'utente. Ad ogni modulo corrispondono dei comandi eseguibili. Ogni comando si comporta in maniera diversa in base ai parametri di ingresso che gli vengono passati.

Ogni operazione, quindi, rappresenta una applicazione univoca del comando di un modulo.

BASE

Il modulo principale dell'app. Contiene le funzionalità di serializzazione e deserializzazione dei dati.

Comandi

SERIALIZE

PARAMETRI RICHIESTI

STRUCTURE

il codice univoco della struttura da usare per generare il dato serializzato deve corrispondere a un codice struttura nella tabella corrispondente e la serializzazione sarà fatta in xml o json in base alla configurazione lì presente.

LINKEDRECORDID

Il record dal quale recuperare i dati per generare il dato serializzato. La tabella deve essere coerente con quanto specificato nella struttura passat~~o~~a da parametro.

SERIALIZEALL

PARAMETRI RICHIESTI STRUCTURE

il codice univoco della struttura da usare per generare il dato serializzato deve corrispondere a un codice struttura nella tabella corrispondente e la serializzazione sarà fatta in xml o json in base alla configurazione lì presente.

DESERIALIZE

PARAMETRI RICHIESTI STRUCTURE

il codice univoco della struttura da usare per interpretare il dato ricevuto deve corrispondere a un codice struttura nella tabella corrispondente e la deserializzazione sarà fatta partendo da un xml o da un json in base alla configurazione lì presente.

INPUTDATA

Deve contenere il dato da deserializzare.

REPORT

È un modulo dedicato alla generazione di report in formato word pdf o Excel. Il modulo può essere evocato sia via API (e in tal caso il report viene restituito in base64 come output) o evocato internamente e il dato viene quindi salvato.

Comandi

PDF

Genera il report in formato PDF.

PARAMETRI RICHIESTI REPORTNO

Il numero di report da eseguire.

LINKEDRECORDID

Se questo campo è valorizzato in una operazione il sistema esegue il report filtrandolo su un singolo record.

INPUTDATA

Può contenere la configurazione da passare alla REQUESTPAGE del report. ES:

<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement" id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1) SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1) SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>

WORD

Genera il report in formato word se il report lo consente.

REPORTNO

Il numero di report da eseguire.

LINKEDRECORDID

Se questo campo è valorizzato in una operazione il sistema esegue il report filtrandolo su un singolo record.

INPUTDATA

Può contenere la configurazione da passare alla REQUESTPAGE del report. ES:

<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement" id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1) SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1) SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>

EXCEL

Genera il report in formato Excel se il report lo consente.

REPORTNO

Il numero di report da eseguire.

LINKEDRECORDID

Se questo campo è valorizzato in una operazione il sistema esegue il report filtrandolo su un singolo record.
INPUTDATA

Può contenere la configurazione da passare alla REQUESTPAGE del report. ES:

<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement" id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1) SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1) SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>

REST

Questo modulo si occupa di creare in base alla parametrizzazione delle operazioni di effettuare delle chiamate rest verso servizi esterni.

Comandi

SEND

PARAMETRI RICHIESTI URL

L'url del servizio remoto.

METHOD

HttpMethod da usare per effettuare la connessione al servizio remoto.

CONTENT-TYPE

Solo se il metodo della chiamata prevede che ci sia un corpo del messaggio va definita la tipologia di dato che si sta trasmettendo.

ES: application/json,application/xml,application/octet-stream

PARAMETRI OPZIONALI

È possibile inserire per le chiamate rest dei parametri opzionali di sottotipo HEADER che verranno aggiunti alla chiamata.

INPUTDATA

Il corpo del messaggio da inviare a destinazione se l'http method associato lo richiede. Il dato deve essere coerente con il content-type indicato nel parametro previsto.

AUTENTICAZIONE

L'autenticazione non è a livello di singola operazione ma a livello di ricetrasmettitore. Le informazioni relative alla autenticazione REST verso un servizio esterno vanno inserite nella sezione info dei parametri in modo da non impattare su quelli relativi alle operazioni.

PARAMETRI OBBLIGATORI

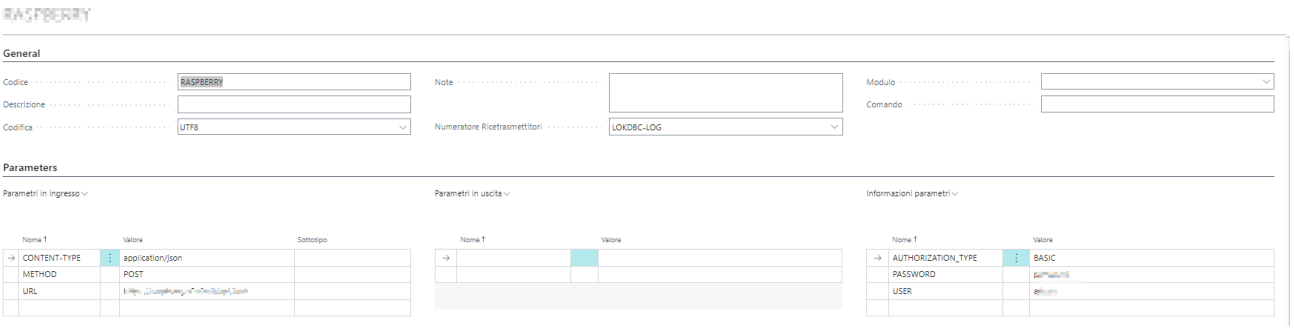
AUTHORIZATION_TYPE

Nativamente il sistema supporta l'autenticazione BASIC e quella OAUTH2 (oltre ovviamente alle chiamate anonime).

BASIC

L'autenticazione BASIC richiede la presenza di due semplici parametri. USER e PASSWORD

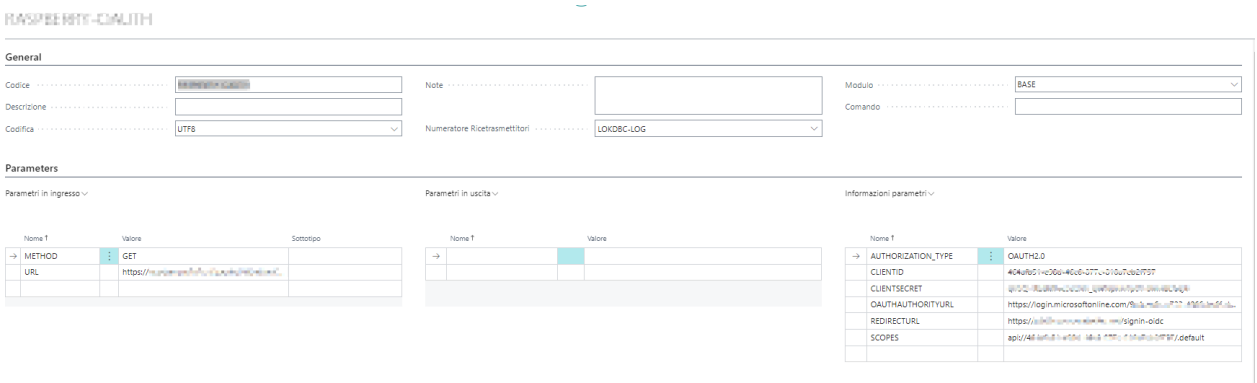
OAUTH2



L'autenticazione OAUTH2 è più complessa e richiede tutti i parametri necessari per ottenere un token.

CLIENTID CLIENTSECRET OAUTHAUTORITYURL REDIRECTURL SCOPES

I parametri seguono lo standard OAUTH2 e non è nello scopo di questo manuale dettagliarne le modalità di compilazione.



TRASFORMAZIONE

Introduzione

Sistema di trascodifica modulare per manipolare un dato prima della serializzazione/-deserializzazione senza vincolarsi al singolo campo.

L'idea alla base di questo sistema di trascodifica è che spesso molteplici strutture ricevute possono necessitare della stessa tipologia di traduzione in fase di presa in carico del dato. L'obiettivo era quello di svincolarsi dal dover ripetere lo stesso elenco di coppie ogni volta.

Ad esempio, la trascodifica di un paese è necessaria all'interno delle anagrafiche clienti ma anche nelle testate dei documenti di vendita o di acquisto e nelle spedizioni. Sono tutte tabelle nel gestionale diverse.

Modello	Tipo Operazione	Tipo di trasformazio...	Da	A
COUNTRY	Ricevi	Semplice	Argentina	AG
COUNTRY	Ricevi	Semplice	Regno unito	UK
COUNTRY	Ricevi	Semplice	Italia	IT
COUNTRY	Invia	Semplice	IT	Italia
COUNTRY	Invia	Semplice	UK	Regno unito
COUNTRY	Invia	Semplice	AG	Argentina

Associando la funzione di trasformazione ad un modello, questo è applicabile in tutte le evenienze in cui la tipologia di dato sia la stessa.

In questo momento il tipo di trasformazione previsto è solo quello semplice: uno scambio 1 a 1 tra un valore in ingresso e uno in uscita distinto per la direzione del flusso dell'operazione ma la funzionalità è estendibile per usare ad esempio regular expression o formule matematiche (ad esempio un dato potrebbe necessitare di una decurtazione percentuale fissa). Eventualmente una trasformazione potrebbe essere applicabile anche a un insieme di record.

Campi

Modello

Il modello dell'insieme di trasformazioni da applicare sulla proprietà della struttura.

Tipo Operazione

Se è una operazione di generazione del dato partendo da business central o piuttosto un invio.

Tipo di trasformazione

In questo momento solo Simple rappresenta uno scambio uno a uno tra il valore Da e il valore A.

Da

Valore di partenza della trasformazione.

A

Valore tradotto.

API

L'intero sistema prevede una singola API di accesso che permette di uniformare, unificare e semplificare ogni comunicazione ricevuta dall'esterno.

http://[URL]/[SERVIZIO]/api/lookie/dataBridge/v2.0/companies([IDCOMPANY])/logs?tenant=[TENANT]

Le parti variabili dell'interfaccia sono l'URL, il TENANT, il nome del SERVIZIO e l'ID della società verso la quale si vogliono indirizzare le chiamate.

Tutte le chiamate verso questo indirizzo dovranno esclusivamente essere in POST e inviare nel corpo del messaggio un contenuto di tipo application/json così strutturato.

```
{

  "transmitterCode": "",
  "module": "",
  "command": "",
  "input": "",
  "parameters": [
    {
      "name": "",
      "value": ""
    }
  ]
}
```

transmitterCode

è il codice identificativo del mittente della chiamata. Deve corrispondere a un record nella tabella trasmittenti o la connessione verrà rifiutata.

module

rappresenta il modulo che espone il comando da eseguire. Se non ci sono altri moduli installati oltre a questo modulo base non è necessario passarlo.

command

rappresenta il comando che si vuole eseguire all'interno del modulo richiesto.

input

è l'oggetto della comunicazione vera e propria e può contenere qualsiasi tipo di dato da un xml a un json a una stringa base64. Starà al codice collegato al command elaborarlo di conseguenza.

parameters

A queste proprietà principali è possibile e a volte necessario aggiungere una serie di parametri relativi al comando richiesto.

I vari parametri saranno passati come coppia name value.

ESEMPIO CHIAMATE API

SERIALIZE

```
{

    "transmitterCode": "VSCODE",
    "command":"SERIALIZE", "input":""
    ,
     "parameters":[
         {
              "name":"STRUCTURE",
```

Questa chiamata effettua con il mittente VSCODE un comando di tipo SERIALIZE. Questo comando richiede un parametro tipo STRUCTURE che indica la struttura che deve essere utilizzata per estrarre i dati.

Il risultato di questa operazione è


```
{

  "@odata.context":
"http://[URL]:7048/[SERVIZIO]/api/lookie/dataBridge/v2.0/$metadata#companies([IDCOMPANY])/logs/$entit y",

  "@odata.etag": "W/\"JzIwOzEzODUyMjgxMjI2MjA2NjI1NzYzMTswMDsn\"",
  "no": "LOG-0000011868",

  "transmitterCode": "VSCODE",
  "command": "SERIALIZE",
  "module": "BASE",

  "input": "",
  "output":

"[{\"ProductID\":\"1\",\"ProductName\":\"Chai\",\"genprodpostinggroup\":\"VARIE\",\"vatprodpostinggroup\":\"IV
A20\",\"inventorypostinggroup\":\"MAT.
PRIME\"},{\"ProductID\":\"10\",\"ProductName\":\"Ikura\",\"genprodpostinggroup\":\"VARIE\",\"vatprodpostingg
roup\":\"IVA20\",\"inventorypostinggroup\":\"MAT.
```


Ai dati della chiamata viene aggiunt~~o una proprietà output contenendo~~a una proprietà output contenente il risultato dell'esecuzione del comando.

Il parametro di ingresso input può essere valorizzato per il comando SERIALIZE con il valore dei filtri da impostare sulla tabella.

```
{

    "transmitterCode": "VSCODE",
    "command":"SERIALIZE", "input":"WHERE(No.=FILTER('LS*'))" ,
     "parameters":[
         {
              "name":"STRUCTURE",
              "value":"ITEM"
```

In questo caso, ad esempio, la tabella associata alla struttura ITEM viene filtrata applicando il filtro LS* al campo "No."

DESERIALIZE

Ritorniamo sull'esempio ~~id~~di deserializzazione usato nella sezione Esempi Struttura.

Il json usato come esempio di deserializzazione in quel contesto andrebbe passato in questa modalità.

```
{

    "transmitterCode": "VSCODE",
    "command":"DESERIALIZE",
    "input":"{
        \"ProductID\": \"1000C\",
        \"ProductName\": \"Bicicletta\",
        \"genprodpostinggroup\": \"VARIE\",
        \"vatprodpostinggroup\": \"IVA20\",
        \"inventorypostinggroup\": \"MAT. PRIME\"
        }
" ,
```

Il risultato ottenuto dalla chiamata è l'inserimento di un nuovo record nella tabella collegata alla struttura ITEM.

CODA PROCESSI

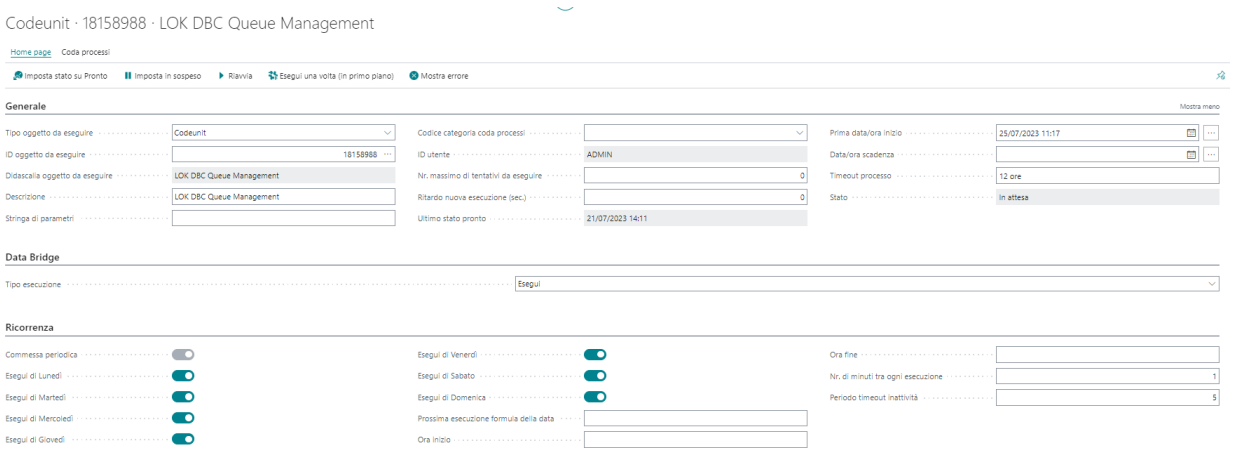
È presente la possibilità di utilizzare una coda processi per l'esecuzione delle operazioni ricevute da API.

Perché una riga ricevuta dalle API non venga immediatamente processata, ma rimanga in stato "Accodato", è necessario che uno dei parametri passati nella chiamata API sia "ASYNC"

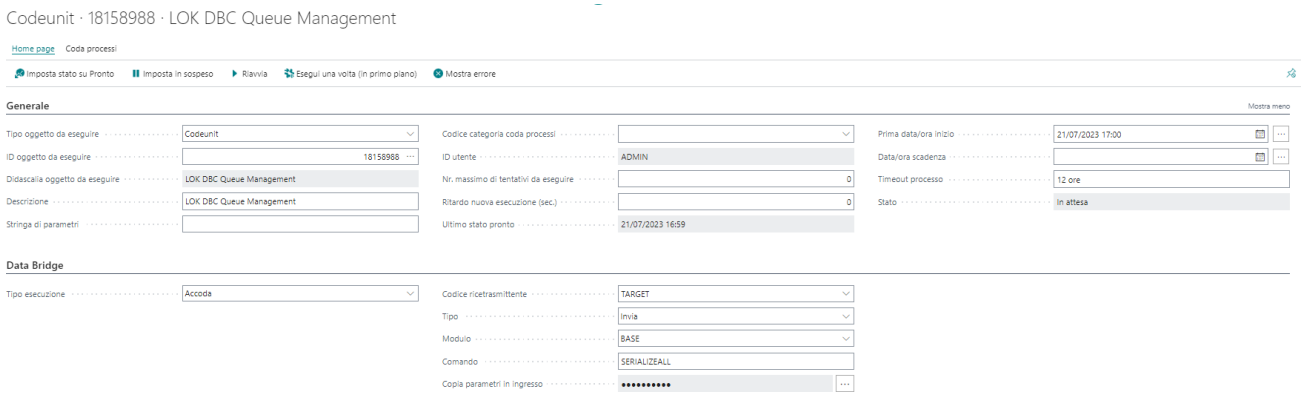
```
{

    "transmitterCode": "VSCODE",
    "command":"DESERIALIZE",
    "input":"{
        \"ProductID\": \"\",
        \"ProductName\": \"Bicicletta\",
        \"genprodpostinggroup\": \"VARIE\",
        \"vatprodpostinggroup\": \"IVA20\",
        \"inventorypostinggroup\": \"MAT. PRIME\"
        }
",
     "parameters":[
        {
```

Con il parametro ASYNC (senza necessità di un value associato) la riga verrà inserita senza essere processata in tempo reale. Al mittente verrà solo restituita l'avvenuta ricezione del dato senza un output che, ovviamente, ancora non esiste. La coda processi è schedulabile fino a un tempo minimo di ricorrenza di un minuto.



La coda processi può anche essere usata per creare in maniera schedulata una nuova riga nel log delle operazioni perché questa venga poi a sua volta processata dal flusso standard.



In questo caso viene richiesto di creare una riga di tipo SERIALIZE-ALL e nelle opzioni parametro viene impostata la struttura da utilizzare.

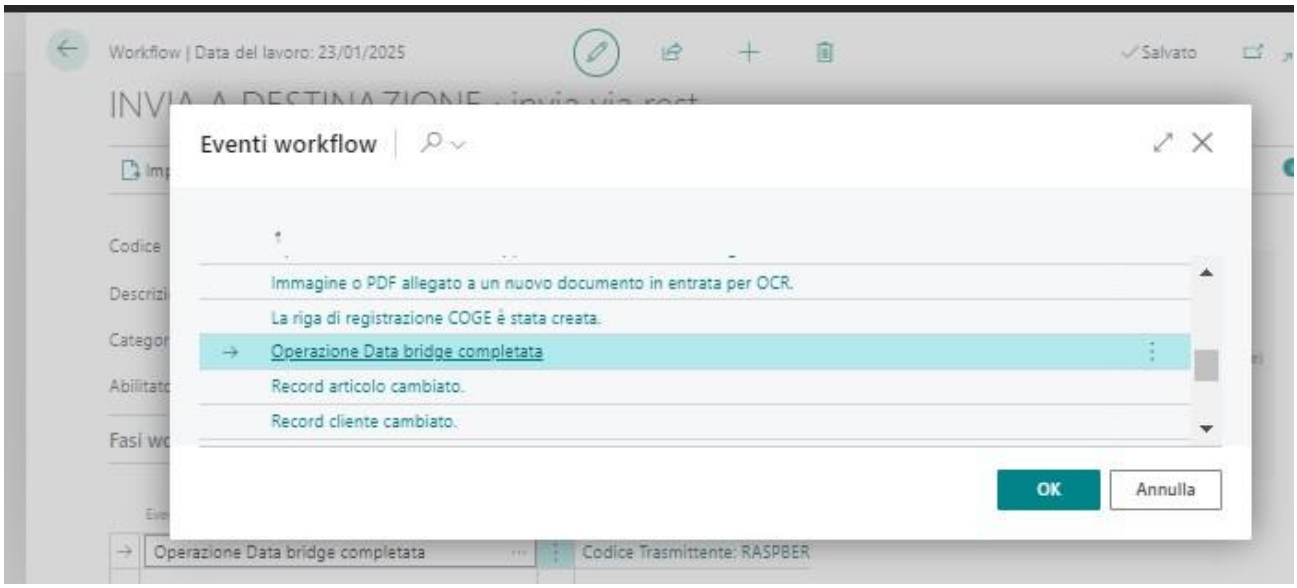
Un utilizzo pratico di questa modalità è, per esempio, quello di gestire in maniera automatica l'invio ad un servizio remoto di tutti i record di una determinata tabella ad intervalli regolari.

Ad esempio, potrebbe essere richiesto di stampare un report riassuntivo e di caricarlo su un sistema documentale esterno.

WORKFLOW

Alla luce delle funzionalità fin qui esposte si è pensato di integrare il sistema delle operazioni visto con i workflow standard di Business Central aggiungendo un evento, una condizione e una risposta.

EVENTO

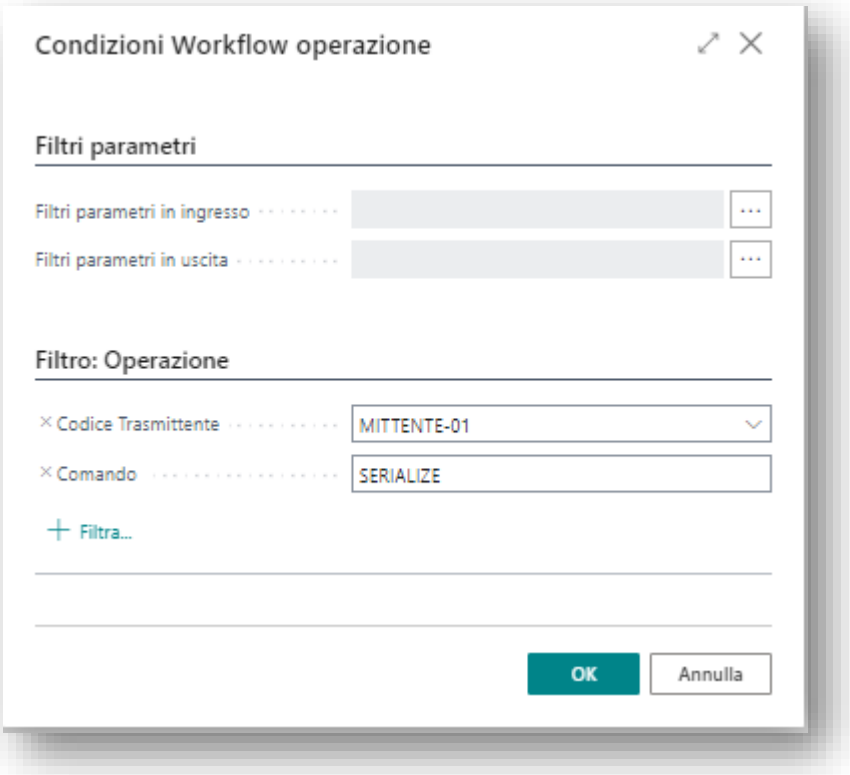


L'evento creato scatta al completamento con successo di una riga operazione nel momento in cui questa passa in stato processato.

È possibile quindi associare un evento concatenato.

CONDIZIONE

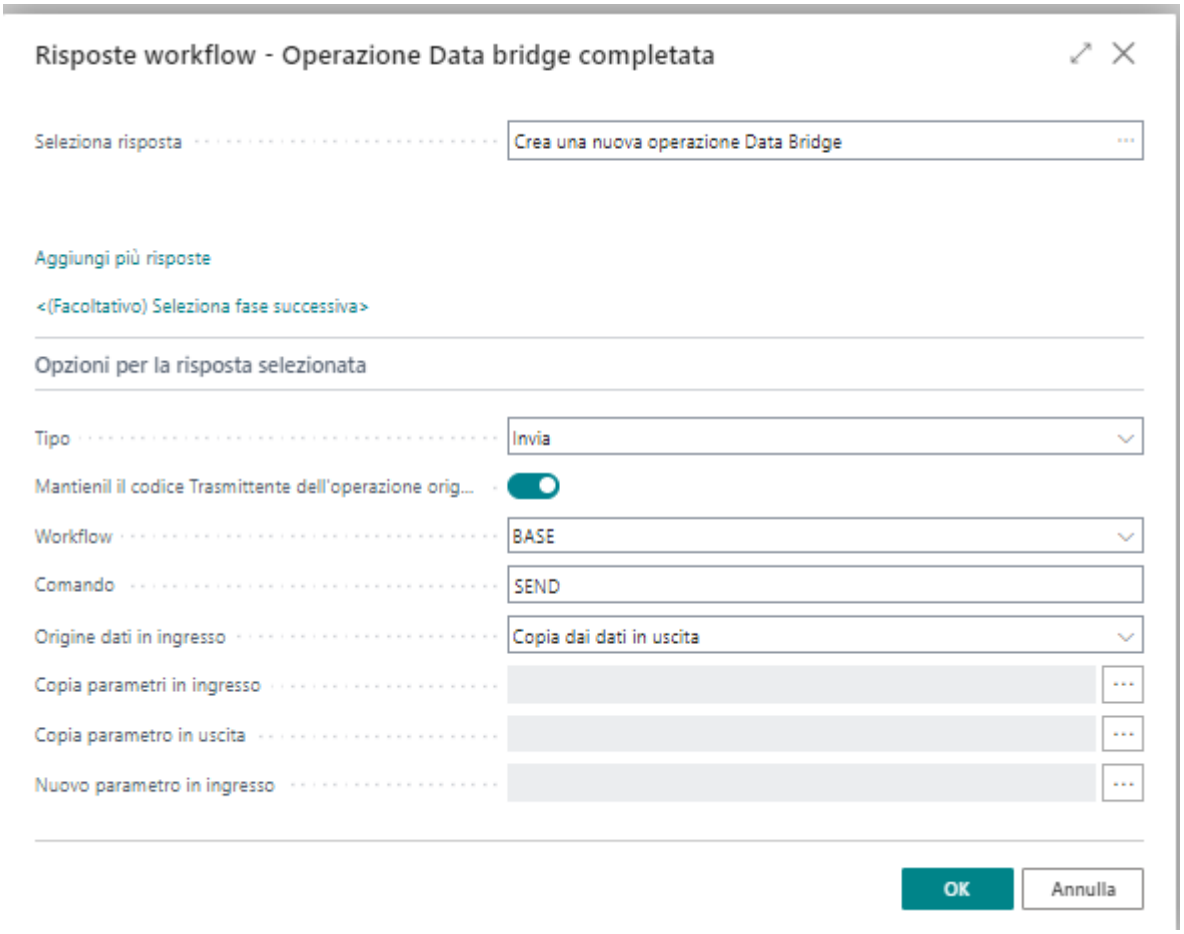
Se l'evento è legato al completamento di una riga operazione la condizione viene modificata non prevedendo soltanto i filtri sul record come da standard ma anche la possibilità di filtrare i parametri in ingresso e in uscita impostando delle coppie nome valore associate.



RISPOSTA

Il sistema prevede la possibilità di creare una riga operazione come risposta a un evento qualsiasi all'interno di un workflow. È possibile impostare dei parametri di ingresso, dei parametri di uscita e, qualora la riga derivi da un evento associato a una operazione, addirittura usare i dati in uscita per primo per passarli al secondo relativamente a parametri e input di comando.

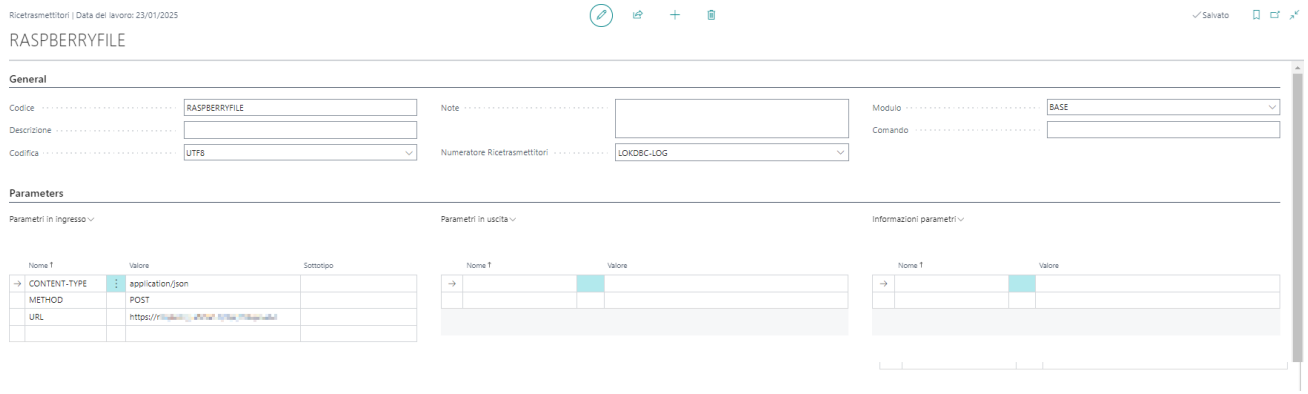
Ovviamente eventuali estensioni di moduli e comandi vengono qui gestiti interamente semplificando di fatto la gestione dei workflow permettendo di creare risposte che eseguono codice complesso senza dover estendere le codeunit standard di gestione dei workflow.



ESEMPIO DI UTILIZZO

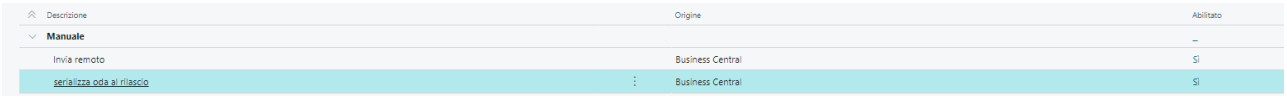
Per il nostro esempio vogliamo inviare a un servizio rest remoto un json contenente le informazioni relative a un ordine di acquisto nel momento in cui questo viene rilasciato.

Per prima cosa definiamo la destinazione di questa chiamata.

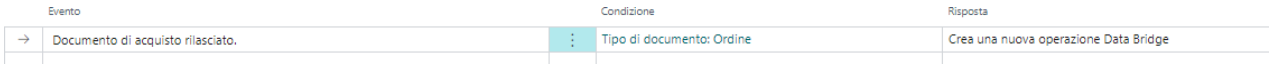


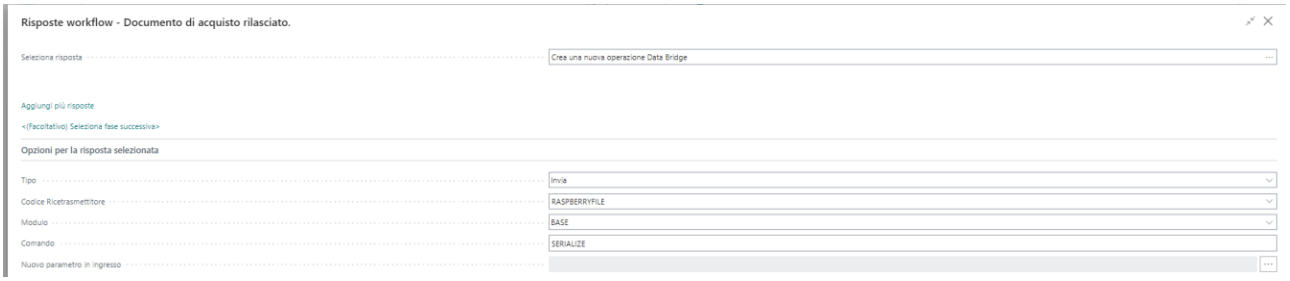
Assegno un URL e un metodo (in questo caso POST).

Nei workflow creo la seguente logica:



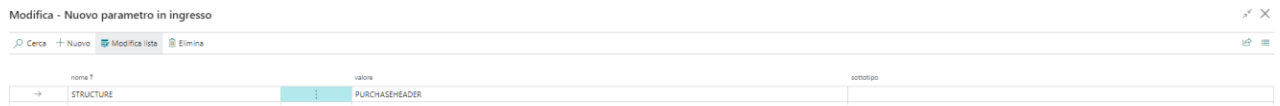
Ogni volta che un ordine di acquisto è rilasciato creo una nuova riga log data bridge.





Una riga che chiede semplicemente di serializzare il dato relativo al documento rilasciato.

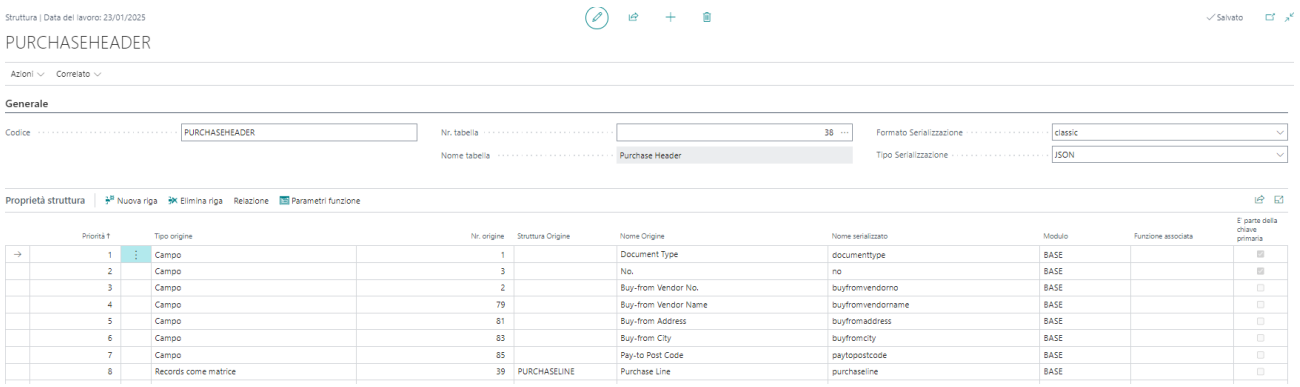
Cliccando su “Edit”[1] accanto alla voce “Nuovo parametro in ingresso” possiamo definire il parametro necessario perché sia possibile serializzare.



Ovvero la struttura da utilizzare “Structure” e assegniamogli il valore libero “Purchaseheader”.

In questa fase è già possibile aggiungere parametri aggiuntivi che potranno essere passati agli step successivi del workflow o anche utilizzati come condizione.

A questo punto è necessario andare a creare una struttura che recuperi i dati dalla tabella 38 e 39 chiamata con lo stesso codice.



Struttura | Data del lavoro: 23/01/2025

PURCHASEHEADER

Salvato

Azioni Correlato

Generale

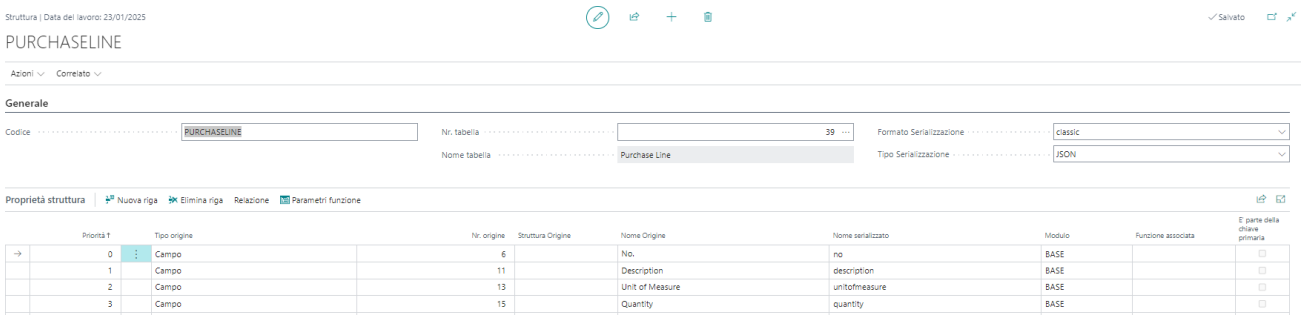
Codice: PURCHASEHEADER | Nr. tabella: 38 | Formato Serializzazione: classic

Nome tabella: Purchase Header | Tipo Serializzazione: JSON

Proprietà struttura — Nuova riga | Elimina riga | Relazione | Parametri funzione

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato	Modulo	Funzione associata	È parte della chiave primaria
1	Campo	1		Document Type	documenttype	BASE		☑
2	Campo	3		No.	no	BASE		☑
3	Campo	2		Buy-from Vendor No.	buyfromvendorno	BASE		☐
4	Campo	79		Buy-from Vendor Name	buyfromvendorname	BASE		☐
5	Campo	81		Buy-from Address	buyfromaddress	BASE		☐
6	Campo	83		Buy-from City	buyfromcity	BASE		☐
7	Campo	85		Pay-to Post Code	paytopostcode	BASE		☐
8	Records come matrice	39	PURCHASELINE	Purchase Line	purchaseline	BASE		☐

E il dettaglio delle righe.



Struttura | Data del lavoro: 23/01/2025

PURCHASELINE

Salvato

Azioni Correlato

Generale

Codice: PURCHASELINE | Nr. tabella: 39 | Formato Serializzazione: classic

Nome tabella: Purchase Line | Tipo Serializzazione: JSON

Proprietà struttura — Nuova riga | Elimina riga | Relazione | Parametri funzione

Priorità	Tipo origine	Nr. origine	Struttura Origine	Nome Origine	Nome serializzato	Modulo	Funzione associata	È parte della chiave primaria
0	Campo	6		No.	no	BASE		☐
1	Campo	11		Description	description	BASE		☐
2	Campo	13		Unit of Measure	unitofmeasure	BASE		☐
3	Campo	15		Quantity	quantity	BASE		☐

Questa configurazione genererebbe un json simile al seguente.

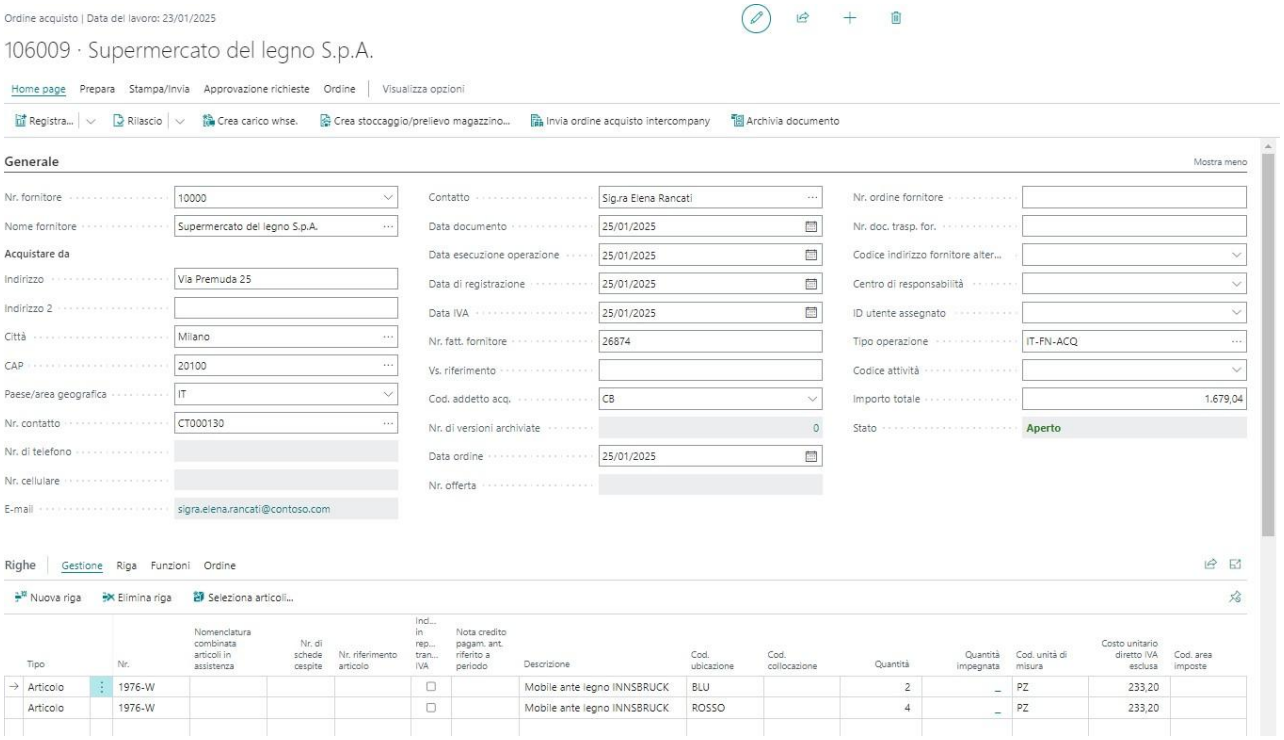

```
{
  "documenttype": 1,
  "no": "104004",
  "buyfromvendorno": "40000", "buyfromvendorname":
  "Sabelli && Fioretti S.r.l.", "buyfromaddress": "Corso
  Colombo 13", "buyfromcity": "Reggio Emilia",
  "paytopostcode": "42100",
  "purchaseline": [
    {
      "no": "LS-10PC",
      "description": "Altoparl., bianco per PC",
      "unitofmeasure": "Scatola",
      "quantity": 40
    },
    {
      "no": "LS-150",
```

```
    "description": "Altop., ciliegia, 150W",
    "unitofmeasure": "Pezzo",
    "quantity": 20
  }
 ]
}
```

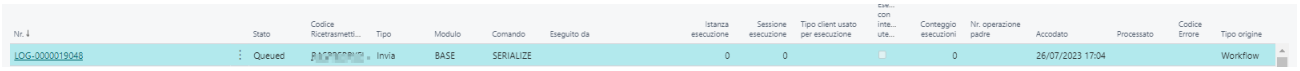
L'anteprima è possibile dal tasto serializza della scheda struttura.

A questo punto, senza aggiungere altro o scrivere codice possiamo rilasciare un ordine di acquisto a caso.

Prendiamo l'ordine

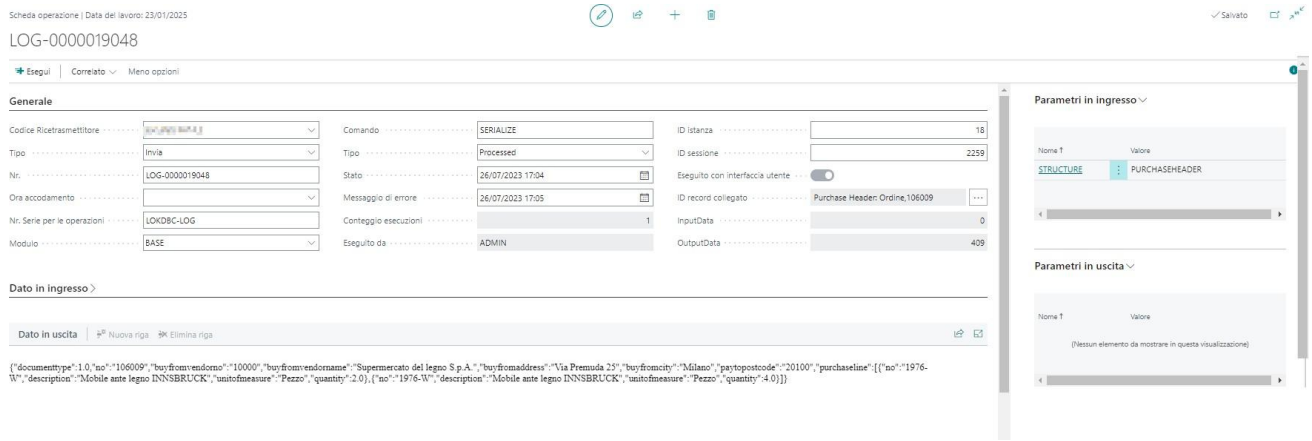


Il rilascio genererà la seguente operazione



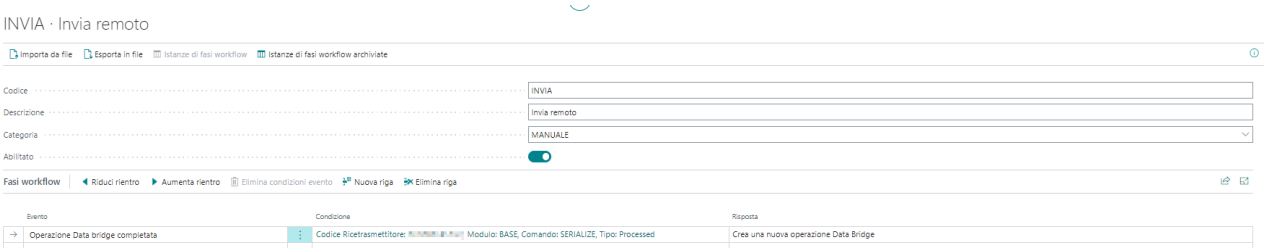
Sappiamo che non è una riga generata da una chiamata api ma da una operazione di workflow e aprendo la scheda abbiamo una serie di dettagli aggiuntivi.

Il record è solo creato in stato attesa e verrà eseguito dalla coda processi schedulata.



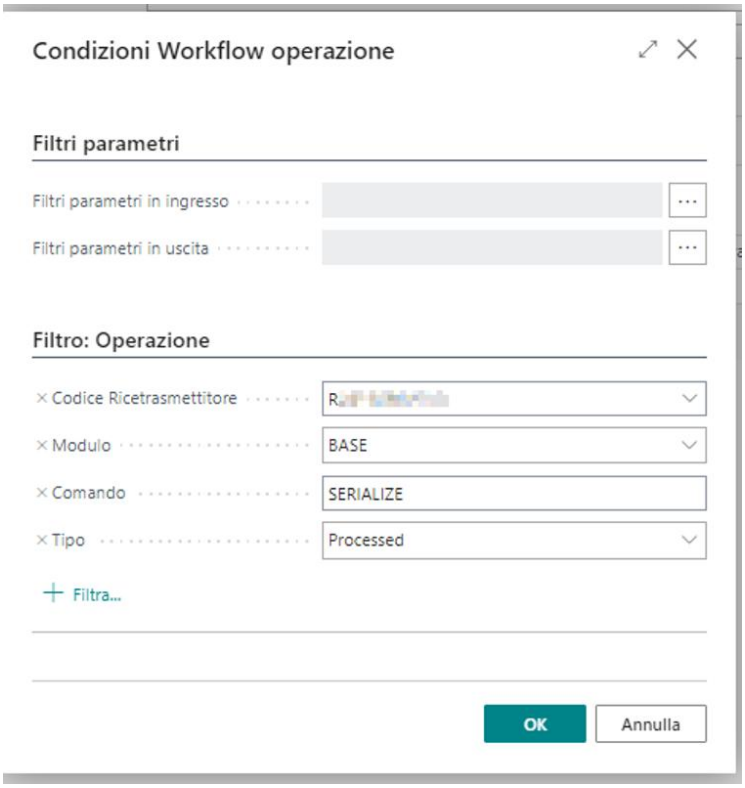
Nel dettaglio vediamo che il parametro struttura utilizzato per la serializzazione è “Purchaseheader” e nei dati in uscita c’è il risultato di questa serializzazione: il json. Sappiamo quale sia il recordid collegato dal quale è possibile accedere velocemente alla scheda del record e una serie di informazioni sulle dimensioni del dato.

A questo punto torniamo nel workflow e aggiungiamo un altro step a questo flusso



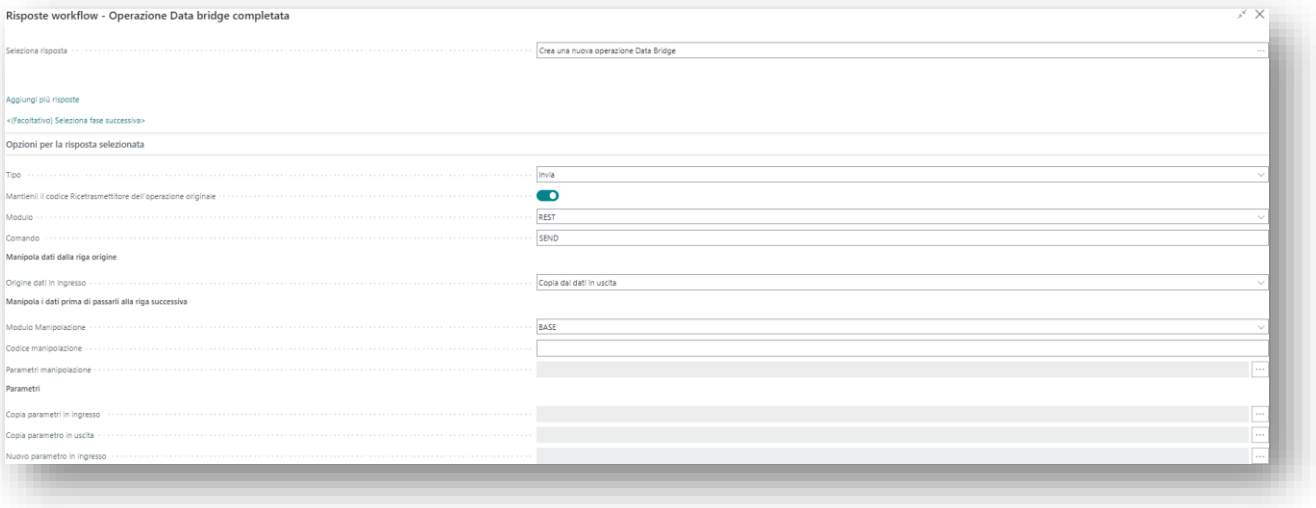
Ogni volta che una riga viene eseguita l’evento viene evocato.

Le condizioni non si limitano ai dati di tabella ma anche ai parametri aggiuntivi



Ad esempio, io potrei definire che questo flusso scatti per quando un record è effettivamente originato da una serializzazione, ma anche aggiungere che è richiesto che il parametro struttura abbia un certo valore.

Questo può servire per definire comportamenti diversi per serializzazioni diverse.



Come risposta a questo evento se si verificano le condizioni chiedo di creare una nuova riga data bridge, di usare i dati di output della prima come ingresso della seconda e di creare una riga che faccia semplicemente un invio rest.

Attivato il workflow riapriamo e rilasciamo lo stesso oda



Che, una volta eseguito genererà



Le due righe risultano collegate dal campo “nr. Operazione padre”. L’esecuzione della chiamata invia il dato a una destinazione remota.



Tutto questo senza scrivere una sola riga di codice.

Allo stesso modo sarebbe stato possibile inviare il layout di un report concatenando una operazione di tipo REPORT a una di tipo REST SEND.

Il sistema, oltre ad avere una serie di estensioni aggiuntive che implementano nuove modalità di comunicazione (sharepoint, file storage, blob storage, mail) è pensato, tramite una serie di eventi ed interfacce perché sia possibile scrivere cod

Contents

Revisions 54
Introduction 55
Extension 55
Technical requirements 55
Dashboard 56
Setup 57
Fields 57
Transceiver No. Series 57
Default Log No. Series 57
Identify the transceiver associated with the current environment 57
Core functions 57
Enable Missing Transformation Error 57
Serialization Filter Type 57
License 57
Expiration Date 57
Expiration Status 57
Actions 57
Omega Group 57
Check License 58
User Setup 58
Transceivers 59
Introduction 59
Fields 59
Encoding 59
No. Series 59
Module 59
Command 59
Parameters 60
Permissions 60
Structures 61
Introduction 61
Fields 61
Code 61
Table No. 61

Table Name 61
Serialization Format 61
Serialization Type 61
Root Element Name and Root Element Namespace 62
Skip OnInsert Trigger 62
Skip OnModify Trigger 62
Actions 62
Add Fields as properties 62
Related 62
Filters 62
Structure Properties 63
Introduction 63
Fields 63
Priority 63
Source Type 63
Source No. 63
Source Structure Code 63
Source Name 64
Serialized Name 64
Module 64
Property Function 64
Is Part of Primary Key 64
Exist Table Relation 64
Exist Table Condition 64
Exist Table Parameter 64
Namespace and isAttribute 64
skipValidate 64
Max String Length 64
Actions 64
Serialize - Deserialize 64
Relations 64
Property Function Parameters 65
StructureCondition 65
Logs 66
Introduction 66
Fields 66

No. 66
Status 66
Transceiver Code 66
Type 66
Module 66
Command 67
Executed By 67
Instance Id 67
Session Id 67
Executed with client type 67
Executed With GuiAllowed 67
Execution Count 67
Parent No. 67
DateTime Queued 67
DateTime Processed 67
Error Code 67
Source Type 67
Input Data 67
Output Data 67
Input Parameters 68
Output Parameters 68
Error Message 68
Callstack 68
Error Codes 69
Introduction 69
Fields 69
Code 69
Regular Expression 69
Log Count 69
Actions 69
Update Logs 69
Dashboard elements in detail 69
Structure examples 71
ITEM SERIALIZATION 71
ITEM DESERIALIZATION 72
SALES ORDER SERIALIZATION 74

FUNCTIONS 81
FIXED 81
Parameters 81
FORMAT 81
Parameters 81
TRANSFORMATION 81
Parameters 81
MODULES 82
CORE 82
Commands 82
REPORT 83
Commands 83
REST 84
Commands 84
Authentication 84
TRANSFORMATION 86
Introduction 86
Fields 86
Model Code 86
Log Type 86
Transformation Type 86
From 86
To 86
API 87
transmitterCode 87
module 87
command 87
input 87
parameters 87
API CALL EXAMPLES 88
SERIALIZE 88
DESERIALIZE 89
Job queue 90
Workflow 92
EVENT 92
CONDITION 93

RESPONSE 94
USAGE EXAMPLE 95

Revisions

Date	Author	Version
11/07/2023	Marco Costanza	Main document
26/07/23	Marco Costanza	Update for version 1.0.21
05/10/2026	Enrico Doretto	Update for version 1.0.22

Introduction

Lookie Data Bridge is the extension designed to simplify the interaction between Business Central and external services, improving at the same time the reception of data and its subsequent processing, and allowing data to be sent to external services through REST calls.

Without writing custom code or creating custom API pages, the extension can be adapted to the exchange format through setup, by configuring the links with the system tables.

This document describes the main configurations an application consultant needs to make the extension operational in its base version.

The application is designed as an extensible modular structure. Installing this extension makes some modules available, each with a set of basic commands that can be used.

Additional modules, which are very simple to create, can be added to extend the system with new tools.

Extension

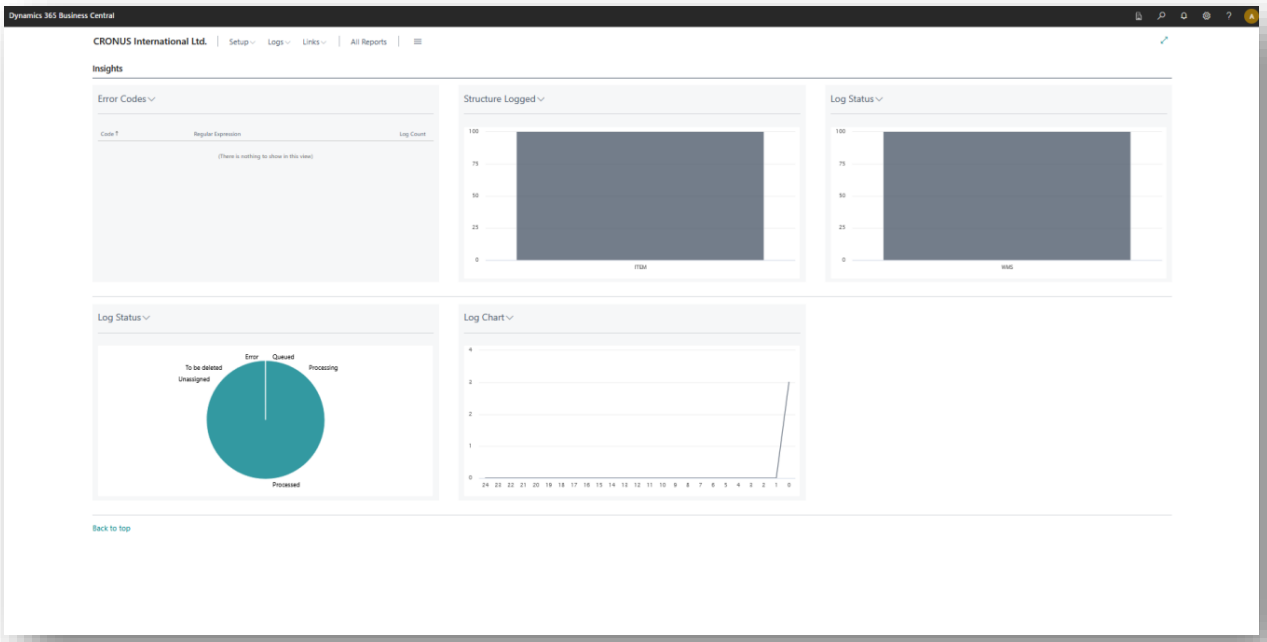
The base module of the Lookie Data Bridge extension has no dependencies and can be installed on any version of Microsoft Dynamics 365 Business Central from version 21 onwards.

Technical requirements

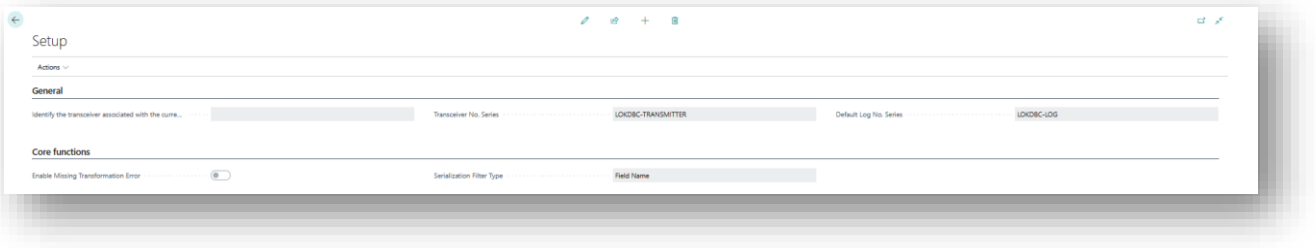
Dashboard

The user responsible for supervising the data exchange to and from the ERP has a dedicated dashboard associated with their user role. The dashboard gives a summary of the status of the data exchanges with the various counterparts in an interactive graphical form, gives access to an aggregated list of the errors found so that they can be solved promptly, and provides a menu with direct access to the various elements.

The dashboard is an essential tool to monitor and efficiently manage the data flow, giving the user a clear and immediate overview of the activities in progress and of the potential issues to address.



Setup



Fields

Transceiver No. Series

The default number series for the transceivers that will be created.

Default Log No. Series

The number series assigned to new transceivers, used to generate the unique log identifiers.

Identify the transceiver associated with the current environment

The transceiver code that identifies the current environment when it exchanges data with another Lookie Data Bridge installation.

Core functions

Enable Missing Transformation Error

If enabled, when a property uses the TRANSFORMATION function and the selected model has no line for the value to transform, the log fails with the error "Invalid transformation filter". If disabled, the value is exported or imported without transformation.

Serialization Filter Type

Defines how the filter passed in the input field of a SERIALIZE command received through the API is interpreted (see section API CALL EXAMPLES).

- Field Name: the filter uses the Business Central field names (e.g. WHERE(No.=FILTER(LS*))).
- Serialized Name: the filter uses the serialized names of the structure properties (e.g. WHERE(ProductID=FILTER(LS*))).

License

Expiration Date

The expiration date of the Lookie Data Bridge license, read from the last license check.

Expiration Status

The license type returned by the Lookie license service in the last check.

Actions

Omega Group

Opens, inside Business Central, the Lookie web page on the Omega Gruppo website (https://www.omegagruppo.it/lookie) with information and contacts.

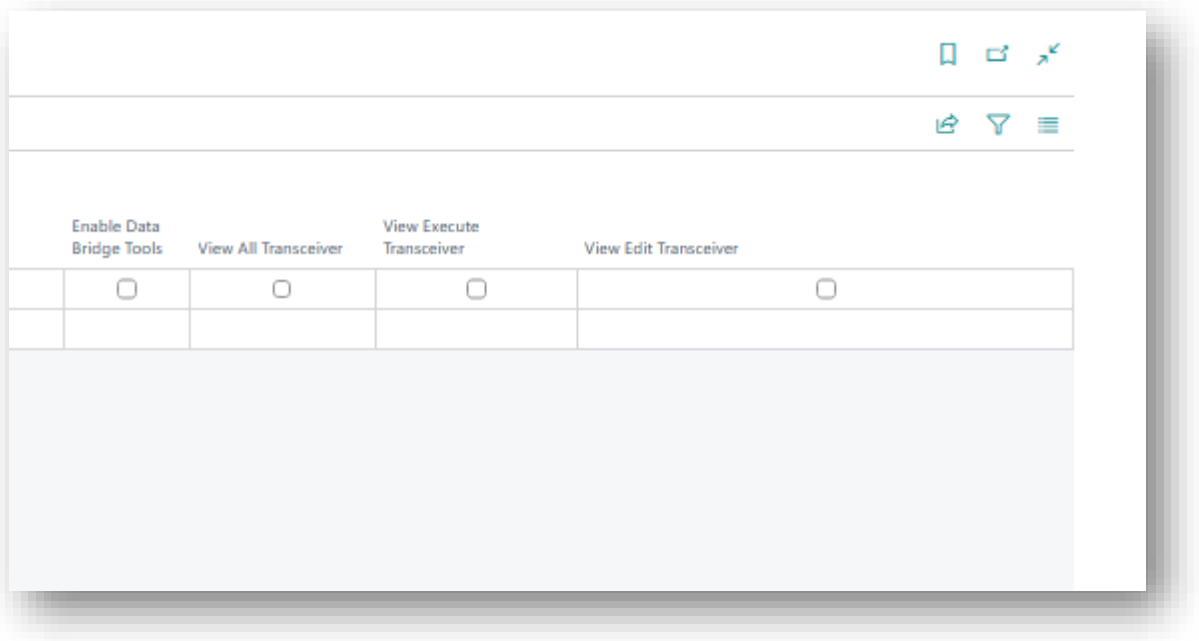
Check License

Contacts the Lookie license service and updates the license data (expiration date and status). If the license is expired or not valid, the error "Invalid License." is shown.

User Setup

Permissions must be set up for a user to use the application.

In User Setup a single user can be given access to all logs, regardless of the transceiver linked to the log.



Viewing, executing and editing the records can be restricted, both for the input data and for the related parameters.

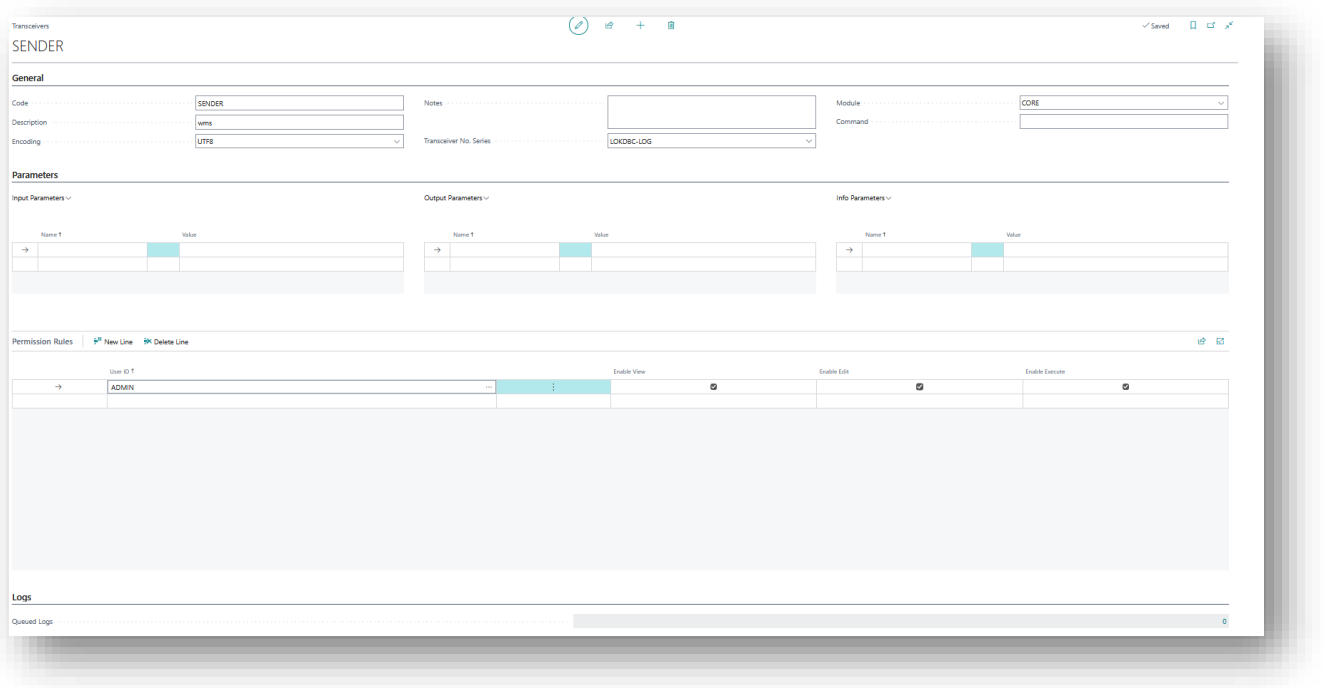
The “Enable Data Bridge Tools” flag activates additional control features, not needed by the application user but useful during development and testing.

If permissions should not be given for the whole application, the features can be restricted per transceiver from the transceiver card.

Transceivers

Introduction

A transceiver is one of the counterparts in a data exchange. Every external service that sends data to Business Central through the features of Lookie Data Bridge must be registered in this table.



Each transceiver has a unique identification number, a description and extended notes.

Fields

Encoding

The Encoding field defines the encoding used for the exchanged data and how the data is stored and handled during its life. The default value is UTF8, but it can be changed to any encoding supported by Business Central.

No. Series

The number series used to generate the unique ID of the logs of the exchanges with this transceiver. New records get the same default number series, but a custom one can be set to have different numbering for the different exchange flows.

Module

The default module of the transceiver, so that it does not have to be passed when data is received. When data is received through the API, the system processes it with this module. Additional modules are added by installing apps that depend on this one and extend its features.

Command

The default command of the transceiver, among those available in the module selected in the “Module” field. When data is received through the API, the system processes it with this command.

The field is free text: the consultant who configures the system must know the correct command to use.

Parameters

Each log has a module and a command. Commands can need parameters to process the log. Parameters are normally set on the single log line, but default parameters can be set on the transceiver card: they are used on every related log unless the log specifies a different value.

There are three types of parameters:

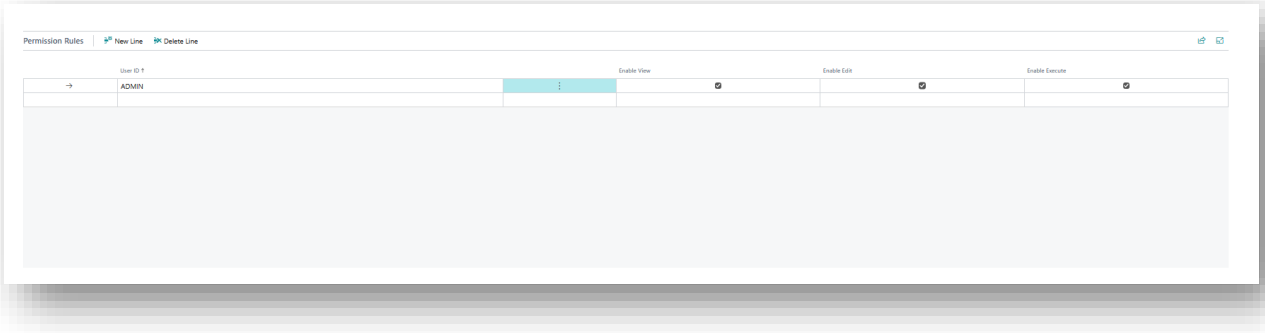
- Input
- Output
- Info

Input parameters are passed to the command of the log.

Output parameters are the result of the log.

Info parameters have no processing effect and are used as notes (for example the REST authentication data).

Permissions



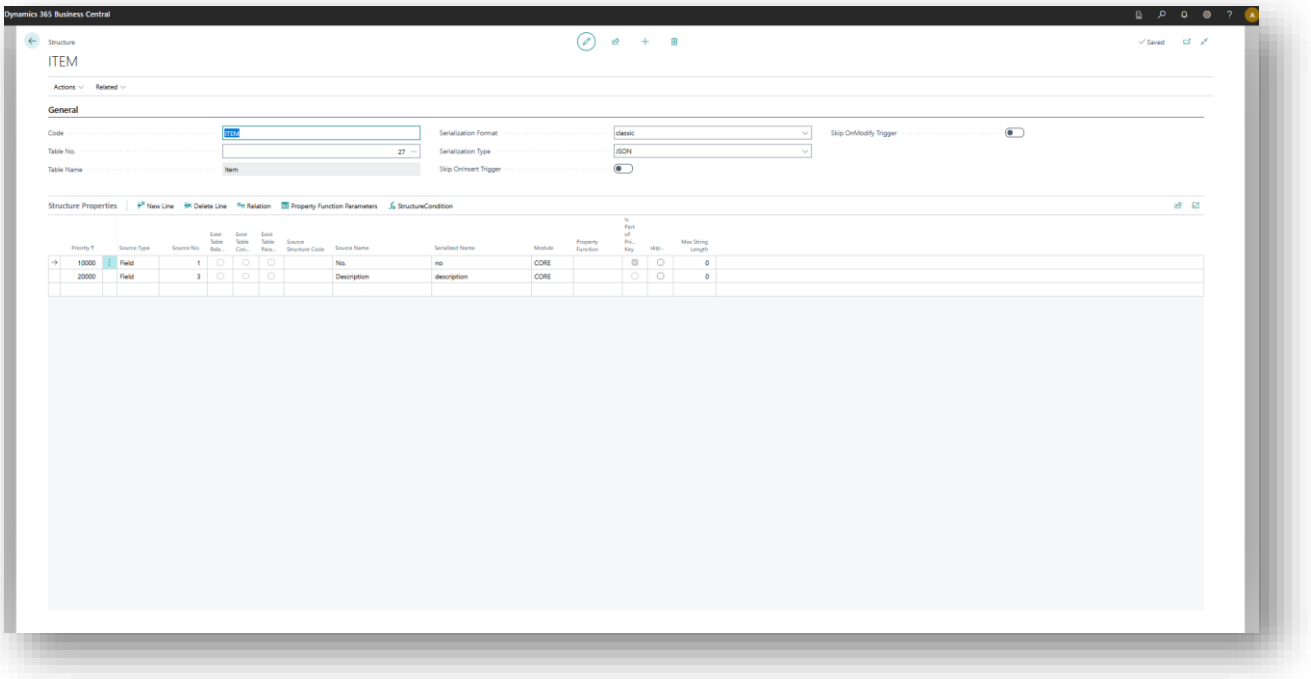
For each transceiver, the permissions to view, execute and edit the data of its logs can be set. These settings also affect the dashboard.

Structures

Introduction

Structures are the exchange entities between Business Central and the external system. They define which information is extracted from the data received by the API and how each piece of information is mapped to the tables and fields of the ERP.

Each structure has a unique code and is linked to a single table.



Fields

Code

The unique code of the structure.

Table No.

The Business Central table the structure is linked to.

Table Name

Lookup field showing the name of the selected table.

Serialization Format

Defines how the serialized names of the properties are built from the name of the field or table. It is a default: every single property name can still be changed manually.

- classic: names are minified (e.g. Sell-to Customer No. → selltocustomerno)
- NAV: names follow the classic NAV naming (e.g. Sell-to Customer No. → Sell_to_Customer_No_)

Serialization Type

The format of the exchanged data: JSON or XML.

Root Element Name and Root Element Namespace

Visible only with XML serialization type: the name and namespace of the root element of the XML document.

Skip OnInsert Trigger

If enabled, during deserialization the new record is inserted without running the OnInsert trigger code of the table.

Skip OnModify Trigger

If enabled, during deserialization the record is modified without running the OnModify trigger code of the table.

Actions

Add Fields as properties

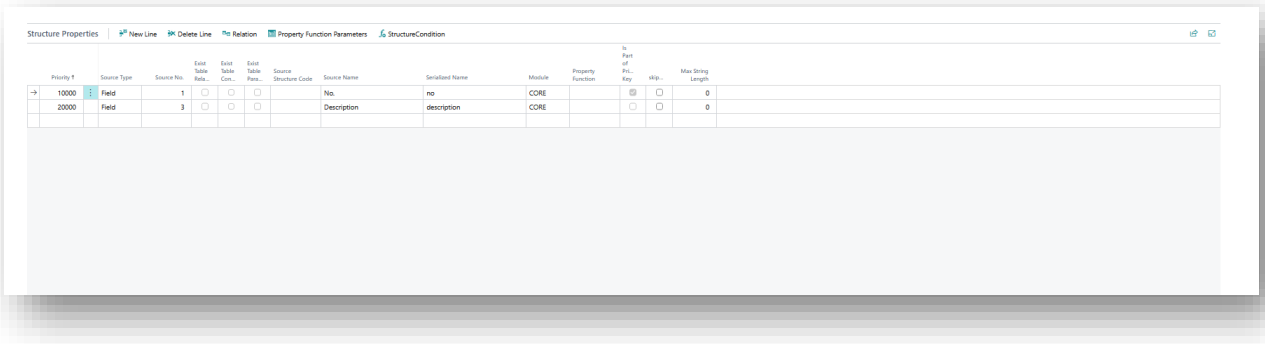
Selects one or more fields of the linked table and adds them in one step as structure properties of type Field.

Related

Filters

Sets fixed filters on the table of the structure. It uses the same page as the relations: for each filter you set the table field, the relation type (CONST or FILTER) and the value. The filters are applied by the SERIALIZEALL command and by the list of records shown by the Serialize action.

Structure Properties



Introduction

Each structure has a set of property lines, each representing an element of the structure. It can be a field of the table itself or another table linked to the main one. In this way multilevel structures can be created that fully describe an object.

Fields

Priority

Defines the processing order of the properties. In serialization it is the order of the properties in the output. In deserialization it is the order in which the fields are validated during import. The order matters because validating a field often fills other fields: to overwrite values set by previous validations, the property that carries the value must be processed later.

Source Type

The type of data the line processes.

- Field: a field of the linked table (e.g. “Sell-to Customer No.” in the “Sales Header” table)
- Record As Array: a related table with several records linked to the record of the parent structure (e.g. “Sales Line” with the “Sales Header” table)
- Record As Object: a related table with a single record linked to the record of the parent structure (e.g. “Payment Method” with the “Sales Header” table)
- Single Property As Value: a related table with a single record linked to the record of the parent structure (e.g. “Payment Method” with the “Sales Header” table), of which only one property is retrieved. It is what a lookup field does in Business Central.
- Inject Values: a related table from which the first record matching the relations is read. Its properties, obtained with the source structure, are added directly to the parent object instead of a nested object; in JSON the name of each property is prefixed with the serialized name of the line. Valid only in serialization.

Source No.

The object the information is retrieved from, based on the source type: a field number or a table number.

Source Structure Code

Valid only for lines of type Record As Object, Record As Array, Single Property As Value and Inject Values: the structure used to retrieve the data from the related record.

Source Name

Name of the field or table set in the Source No. field.

Serialized Name

The name of the serialized element for the object selected in the source.

Module

The module that provides the data manipulation function applied to the property (Property Function field). In the base version the functions are provided by the CORE module.

Property Function

The data manipulation function of the line. See section FUNCTIONS.

Is Part of Primary Key

Calculated field showing whether the line refers to a field that is part of the primary key of the table of the structure.

Exist Table Relation

Calculated field showing whether relations are defined for the line (Relation action).

Exist Table Condition

Calculated field showing whether conditions are defined for the line (StructureCondition action).

Exist Table Parameter

Calculated field showing whether the line has parameters for its property function (Property Function Parameters action).

Namespace and isAttribute

Visible only with XML serialization type. Namespace is the namespace of the XML element of the property; isAttribute serializes the property as an attribute of the parent element instead of a child element.

skipValidate

If enabled, during deserialization the value is assigned to the field without running the OnValidate trigger code.

Max String Length

If set, in JSON serialization text and code values are truncated to this number of characters.

Actions

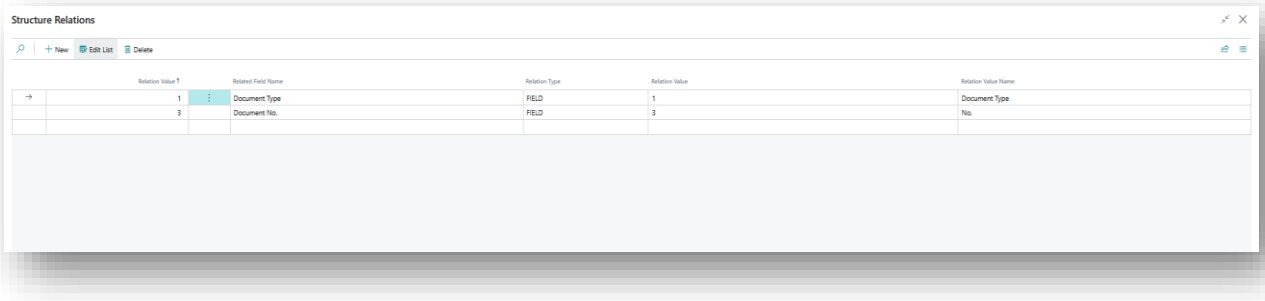
Serialize - Deserialize

Serialize selects a record of the table set in the Table No. field and shows the data serialized according to the structure properties. Deserialize takes input data and creates the record.

See the examples section for details.

Relations

Defines the links between a property line related to another table and the record of the parent structure.



- Relation Value (first column): the field number of the related table used to create the relation with the parent table.
- Relation Type: how the field of the first column is handled: CONST, FILTER or FIELD.
- Relation Value: with relation type FIELD it is the field of the parent table to link to; with CONST it is the constant value used to find the records of the related table; with FILTER it uses the standard Business Central filter format (e.g. I*).

In the example the Sales Line table is linked to the Sales Header table by the “Document Type” field and by the “Document No.” field, which corresponds to the “No.” field of the parent table.

Property Function Parameters

Opens the parameters required by the property function of the line (e.g. VALUE for FIXED, FORMATPROPERTY for FORMAT, MODEL for TRANSFORMATION). The parameters are created automatically when the function is selected; their value must be filled in.

StructureCondition

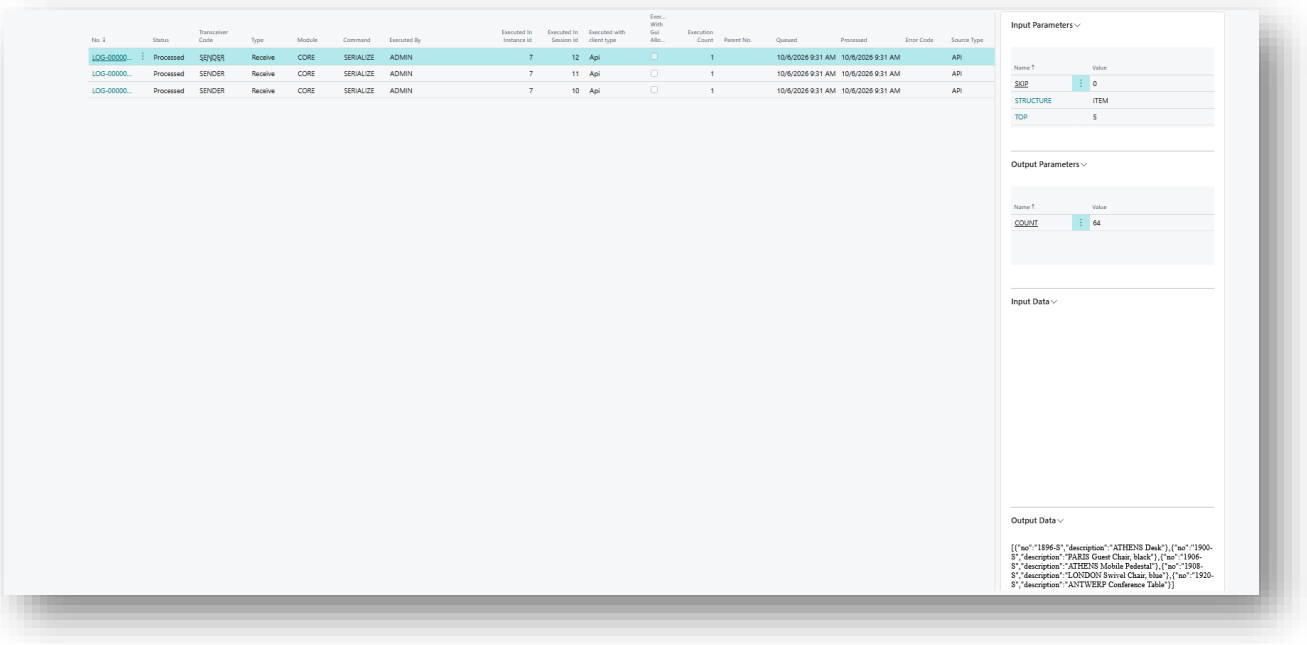
Defines the conditions the record must meet for the property to be included in the JSON serialization. For each condition you set the field of the table of the structure (Related Table No. and Related Field No.), the condition type (CONST or FILTER) and the condition value. If the record does not meet the conditions, the property is not exported.

Logs

Introduction

Every communication sent to the system through the API is recorded in the Logs table. A log is an atomic data operation, based on the transceiver code, the module and the command chosen among those available for the module.

Each log can have additional input parameters used by the chosen command.



Fields

No.

Unique number of the log.

Status

The status of the line. When it is inserted it is “Queued”; after execution it becomes Processed or Error. Other intermediate statuses exist for specific cases.

Transceiver Code

The transceiver code of the log.

Type

- Receive
- Send
- Info
- Detailed

Module

The module of the log.

Command

The command of the log, among those available in the chosen module. The same command code can therefore exist in different modules, leaving more freedom for extensions.

Executed By

The user who executed the log.

Instance Id

The service instance the log was executed on.

Session Id

The session the log was executed in.

Executed with client type

The client type that executed the log.

- Web: from the user interface
- ODataV4: received through OData web services
- Api: call received through the Lookie Data Bridge API
- Background: from the job queue

Executed With GuiAllowed

Shows whether the execution happened with the user interface.

Execution Count

Since logs can be edited and executed several times manually, this field counts how many times the command of the line has been executed.

Parent No.

If the log was created by the processing of another log, this field contains the number of that log.

DateTime Queued

Date and time the line was added to the logs.

DateTime Processed

Date and time of the last execution of the line.

Error Code

The error code assigned to the error message of the log according to the regular expressions of the "Error Codes" table (see section Error Codes).

Source Type

Shows how the line was created: User (inserted by a user), API (received through the API), Workflow (created by a workflow response) or Queue (created by the SERIALIZEALL command).

Input Data

The input data of the log, coming from the input field of the API. Visible in the FactBox of the list and in the log card.

Output Data

The result of the log, returned to the API in the output property. Visible in the FactBox of the list and in the log card.

Input Parameters

The input parameters the command needs to execute the log. Visible in the FactBox of the list and in the log card.

Output Parameters

The parameters resulting from the log. Visible in the FactBox of the list and in the log card.

Error Message

The error message if the execution failed. Visible in the FactBox of the list and in the log card.

Callstack

The call stack of the error if the execution failed. Visible in the FactBox of the list and in the log card.

Error Codes

Introduction

The Error Codes table classifies the failed logs. Each code has a regular expression: the logs whose error message matches the expression get that code. The table is available from the Setup menu of the dashboard and is also shown in the dashboard with the number of logs per code.

Fields

Code

The unique code of the error.

Regular Expression

The regular expression compared with the error message of the logs.

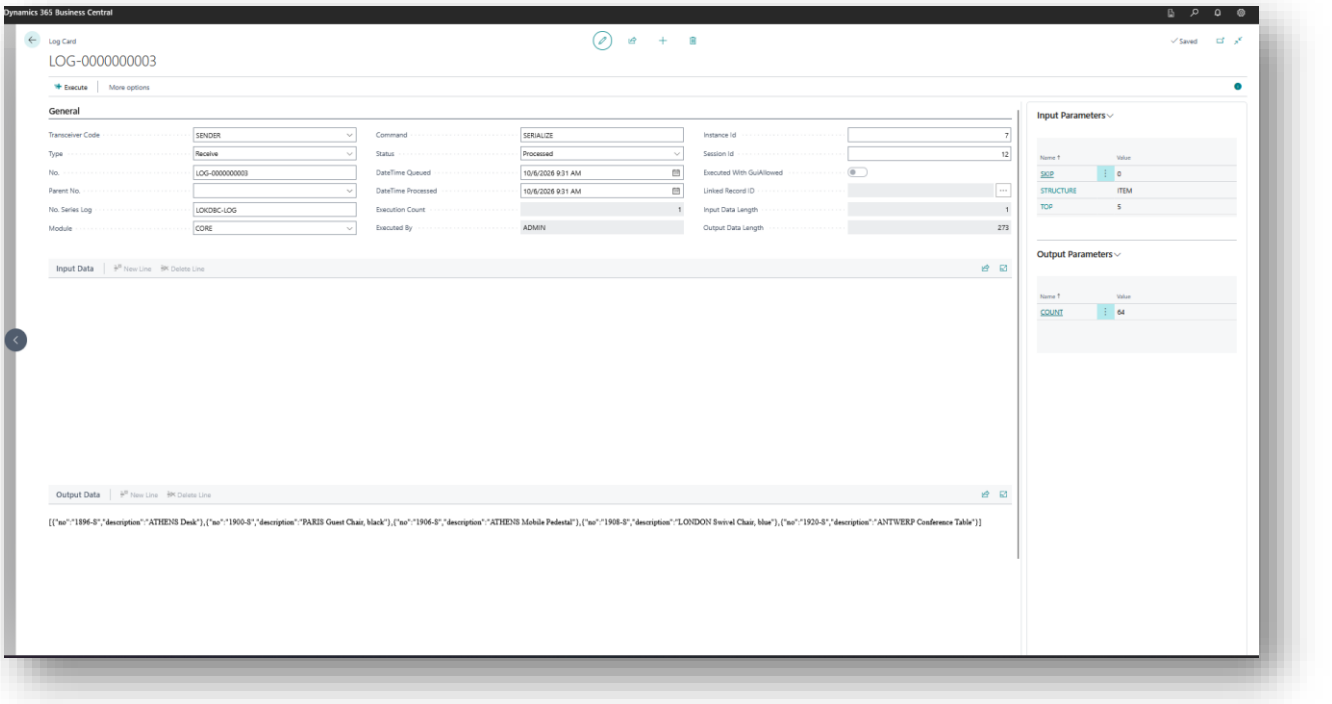
Log Count

Calculated field with the number of logs that have the code.

Actions

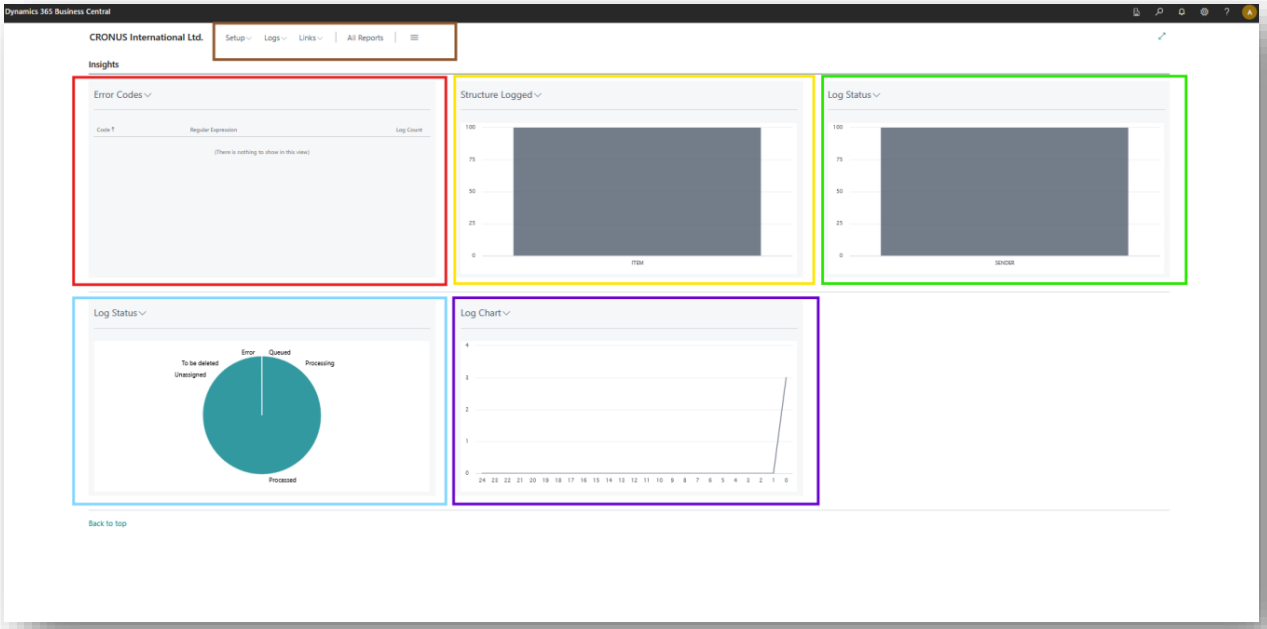
Update Logs

Checks the logs in Error status that have no error code yet and assigns the code whose regular expression matches the error message. The code is not assigned automatically at execution: this action must be run.



Dashboard elements in detail

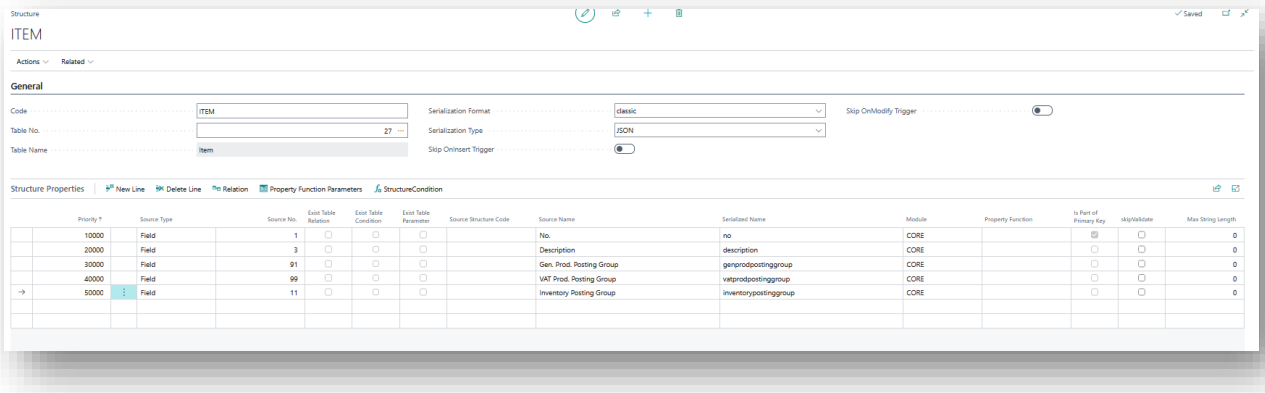
In its default configuration, without personalization, the dashboard shows the following elements.



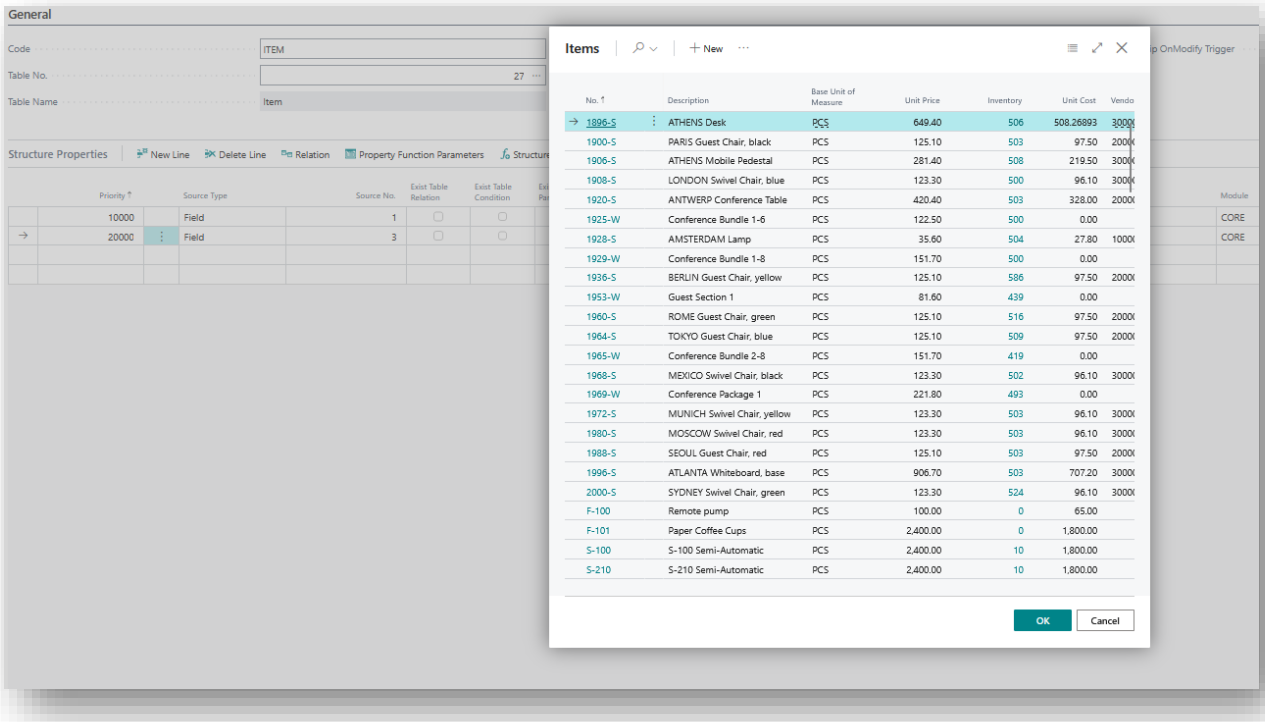
- ■ List of the error codes assigned to the logs.
- ■ Chart of the logs grouped by status and by structure.
- ■ Chart of the logs grouped by status and transceiver code.
- ■ Chart of the logs grouped by status.
- ■ Chart of the logs grouped by transceiver code with the number of logs in the last 24 hours.
- ■ Menu to access the pages.

Structure examples

ITEM SERIALIZATION



Selecting a record of the Item table.



generates the following JSON

```
{
   "ProductID": "1000",
   "ProductName": "Bicycle",
   "genprodpostinggroup": "MISC",
   "vatprodpostinggroup": "VAT20",
   "inventorypostinggroup": "RAW MAT"
}
```

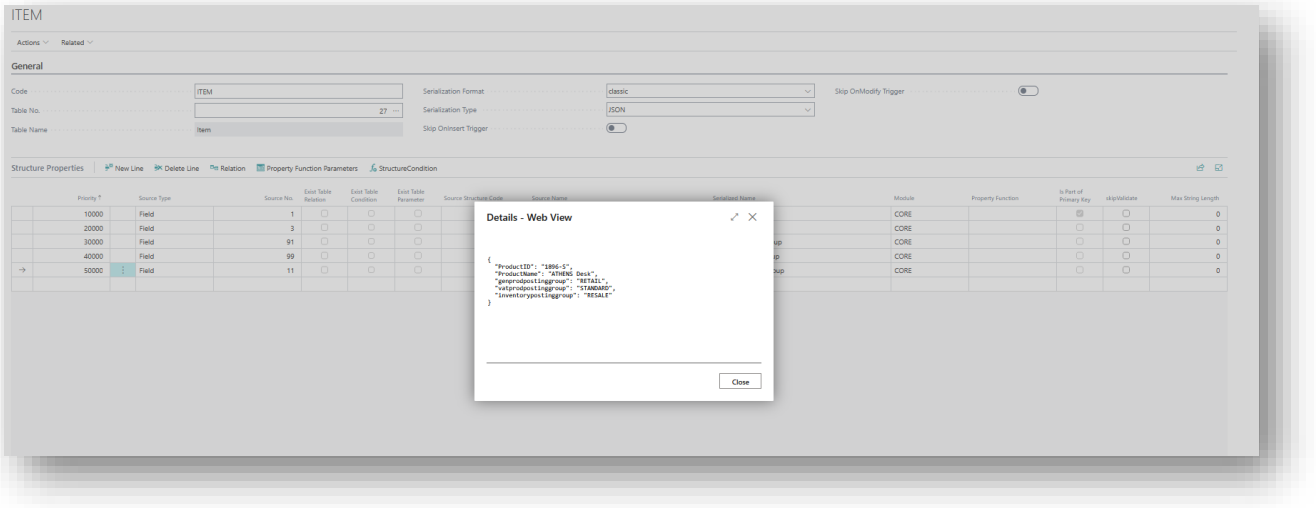
ITEM DESERIALIZATION

Creates a record in the table set in the “Table No.” field, interpreting the data as configured in the properties.

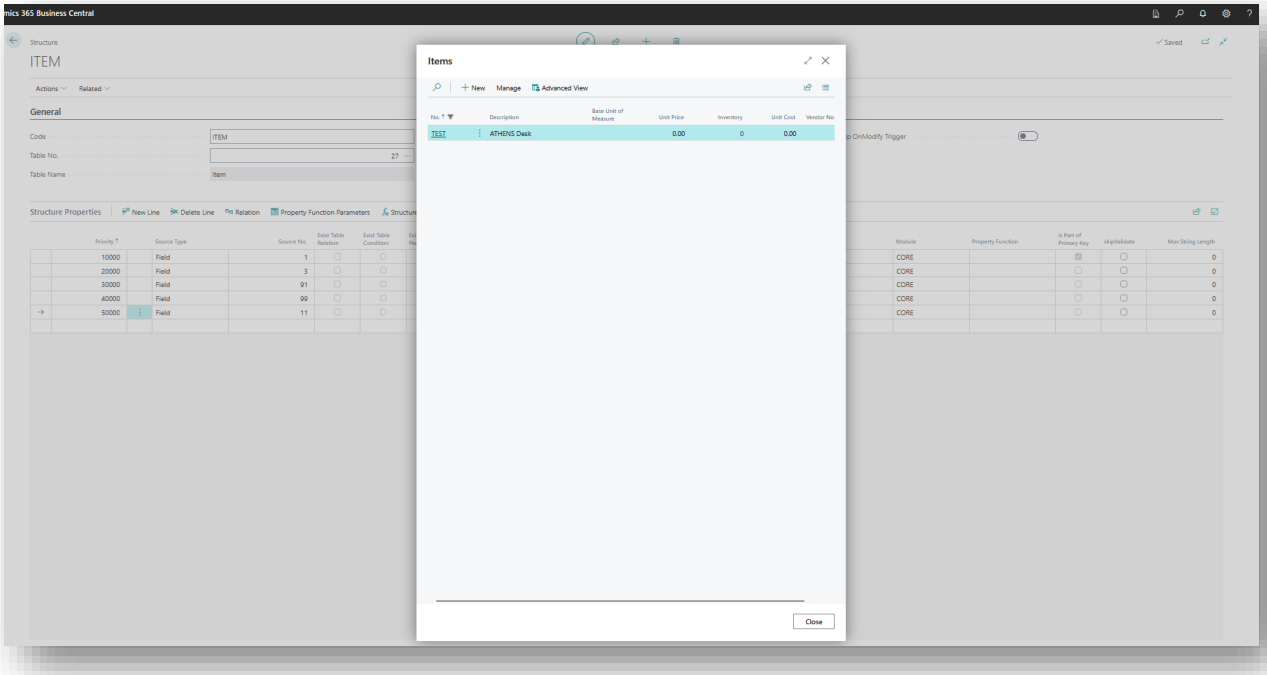
For example, with this JSON as input

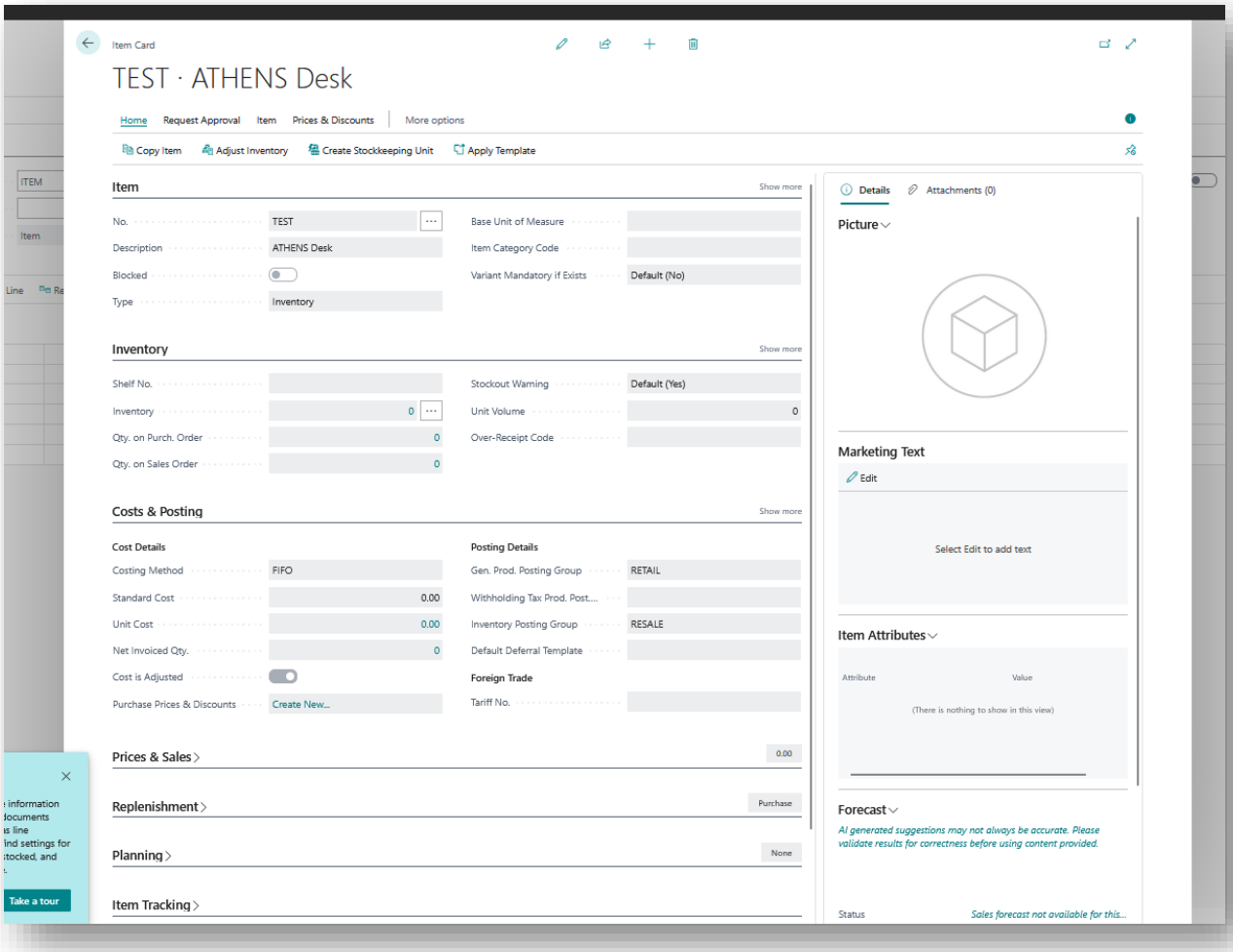
```
{
   "ProductID": "1000B",
   "ProductName": "Bicycle",
   "genprodpostinggroup": "MISC",
   "vatprodpostinggroup": "VAT20",
   "inventorypostinggroup": "RAW MAT"
}
```

Input



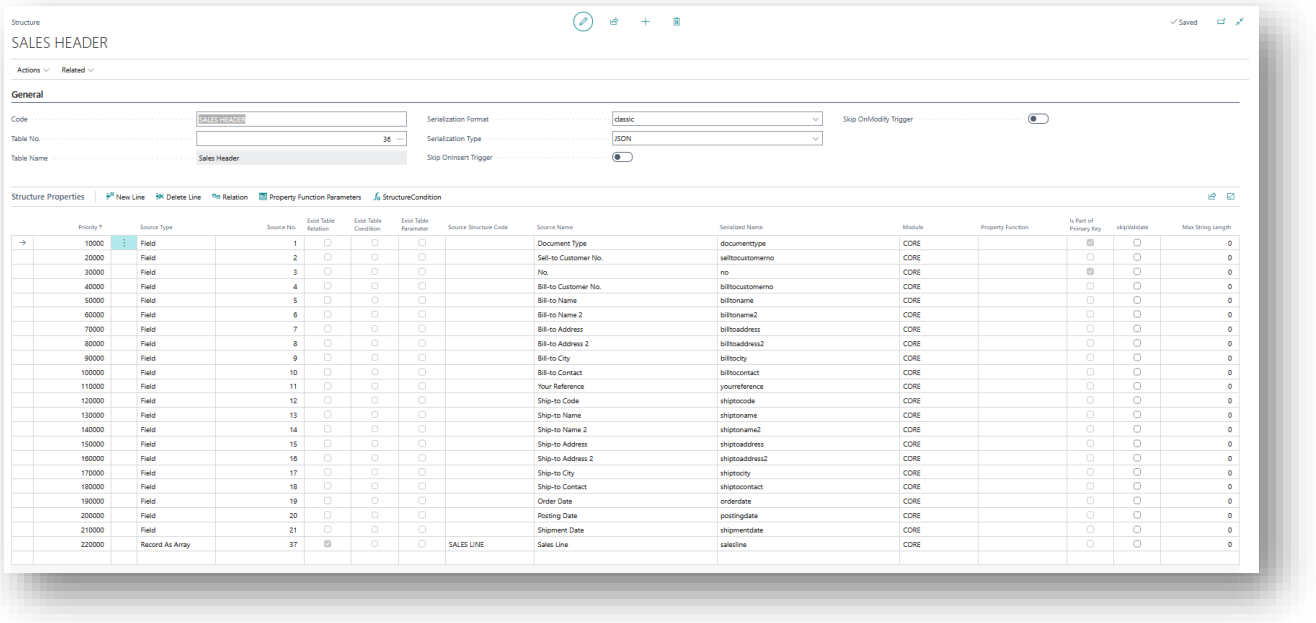
the system creates the record and opens its lookup page.


ITEM
Items
TEST
ATHENS Desk
Close


Item Card
TEST · ATHENS Desk
Home
Request Approval
Item
Prices & Discounts
More options
Copy Item
Adjust Inventory
Create Stockkeeping Unit
Apply Template
Item
No.
TEST
Description
ATHENS Desk
Blocked
Type
Inventory
Base Unit of Measure
Item Category Code
Variant Mandatory if Exists
Default (No)
Inventory
Shelf No.
Inventory
Qty. on Purch. Order
Qty. on Sales Order
Stockout Warning
Default (Yes)
Unit Volume
Over-Receipt Code
Costs & Posting
Cost Details
Costing Method
FIFO
Standard Cost
Unit Cost
Net Invoiced Qty.
Cost is Adjusted
Purchase Prices & Discounts
Create New...
Posting Details
Gen. Prod. Posting Group
RETAIL
Withholding Tax Prod. Post....
Inventory Posting Group
RESALE
Default Deferral Template
Foreign Trade
Tariff No.
Prices & Sales
Replenishment
Purchase
Planning
None
Item Tracking
Details
Attachments (0)
Picture
Marketing Text
Edit
Select Edit to add text
Item Attributes
Attribute
Value
(There is nothing to show in this view)
Forecast
AI generated suggestions may not always be accurate. Please validate results for correctness before using content provided.
Status
Sales forecast not available for this...
Take a tour

SALES ORDER SERIALIZATION

A sales order is used as an example of serialization of a single document with its lines, details taken from the payment terms and the name of the salesperson.



The linked table is 36 Sales Header and the serialization format is classic, so the serialized names are the minified field names.

The detail lines are also fields of the same table.

Priority	Source Type	Source No.	Source Structure Code	Source Name	Serialized Name
0	Field	1		Document Type	documenttype
1	Field	3		No.	no
2	Field	2		Sell-to Customer No.	selltocustomerno
3	Field	19		Order Date	orderdate

To add the sales lines to the structure, a structure property is defined as follows.

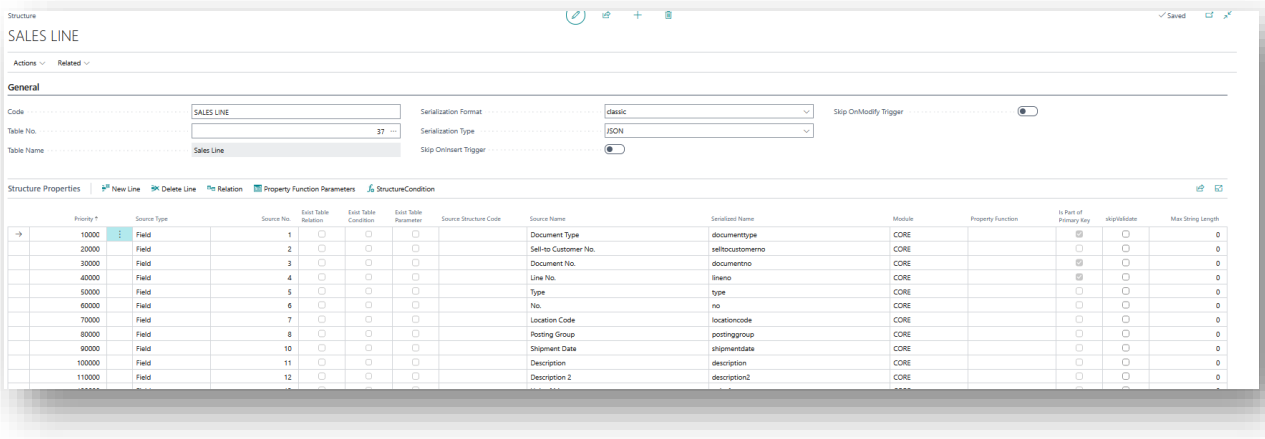
Priority	Source Type	Source No.	Source Structure Code	Source Name	Serialized Name
4	Record As Array	37		Sales Line	salesline

First, define how the details of table 37 are extracted.

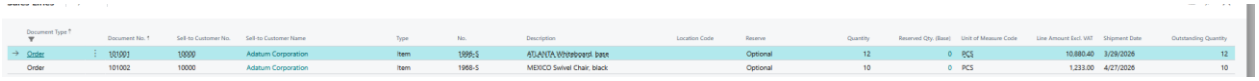
A structure linked to table 37 is defined for the mapping.

The source type is “Record As Array” because several records are retrieved and exposed as an array of objects.

Create the “Sales Line” structure, link it to table 37 and add the properties for the fields to retrieve.



As in the previous item example, serializing a sales document line generates the following output.

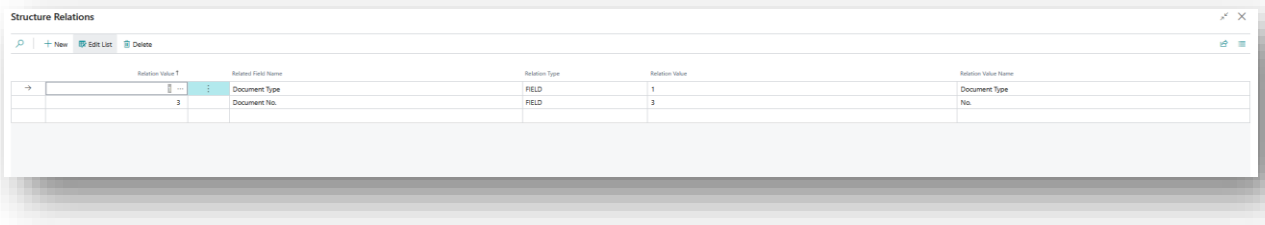


```
{
   "documenttype": 1,
   "selltocustomerno": "10000",
   "documentno": "101001",
   "lineno": 10000,
   "type": 2,
   "no": "1996-S",
   "locationcode": "",
   "postinggroup": "RESALE",
   "shipmentdate": "2026-03-29",
   "description": "ATLANTA Whiteboard, base",
   "description2": "",
   "unitofmeasure": "Piece",
   "quantity": 12,
   "outstandingquantity": 12,
   "qtytoinvoice": 12,
   "qtytoship": 12,
   "unitprice": 906.7,
   "unitcost(lcy)": 707.2,
   "vat%": 25,
   "linediscount%": 0
}
```

Back in the “Order” structure, the new structure can be selected on the line linked to table 37.

Priority	Source Type	Source No.	Source Structure Code	Source Name	Serialized Name
4	Record As Array	37	ORDERLINE	Sales Line	salesline

Finally, set the relations between table 37 and table 36 as described in the Relations section.



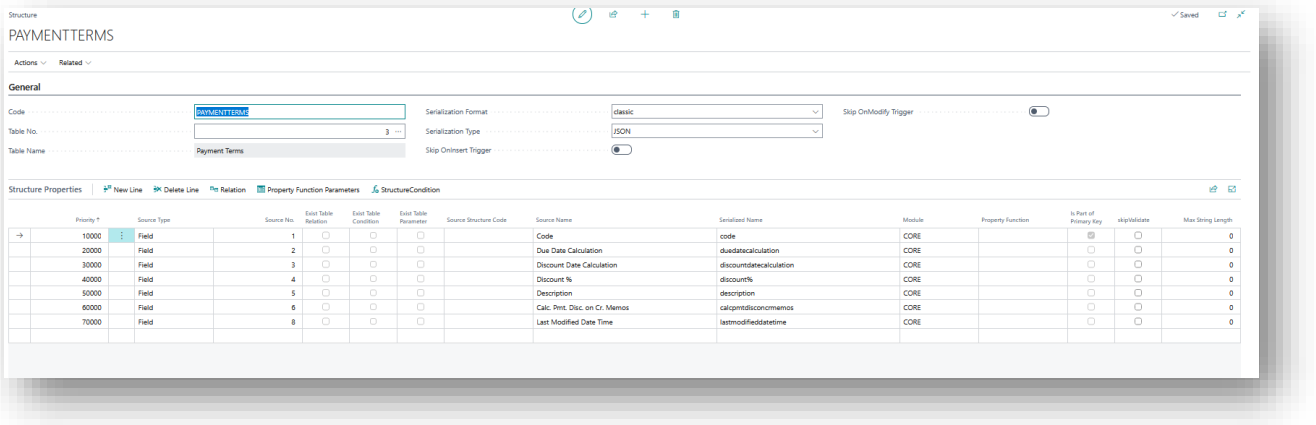
Now the serialization of an order produces the following output.


```
{
   "documenttype": 0,
   "selltocustomerno": "20000",
   "no": "1001",
   "billtocustomerno": "20000",
   "billtoname": "Trey Research",
   "billtoname2": "",
   "billtoaddress": "Southwark Bridge Rd, 91-95",
   "billtoaddress2": "",
   "billtocity": "London",
   "billtocontact": "Helen Ray",
   "yourreference": "OPEN",
   "shiptocode": "",
   "shiptoname": "Trey Research",
   "shiptoname2": "",
   "shiptoaddress": "Southwark Bridge Rd, 91-95",
   "shiptoaddress2": "",
   "shiptocity": "London",
   "shiptocontact": "Helen Ray",
   "orderdate": "2025-01-01",
   "postingdate": "2026-01-31",
   "shipmentdate": "2026-10-01",
   "salesline": [
      {
         "documenttype": 0,
         "selltocustomerno": "20000",
         "documentno": "1001",
         "lineno": 10000,
         "type": 2,
         "no": "1936-S",
         "locationcode": "",
         "postinggroup": "RESALE",
         "shipmentdate": "2026-10-01",
         "description": "BERLIN Guest Chair, yellow",
         "description2": "",
         "unitofmeasure": "Piece",
         "quantity": 10,
         "outstandingquantity": 10,
         "qtytoinvoice": 10,
         "qtytoship": 10,
         "unitprice": 125.1,
         "unitcost(lcy)": 97.5,
         "vat%": 25,
         "linediscount%": 0
      }
   ]
}
```


An order header with its lines.

Add a line for the payment terms.

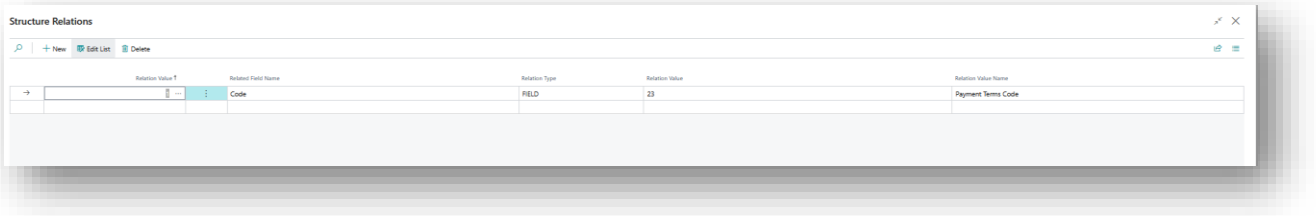
After creating a structure linked to table 3 “Payment Terms”,



a new property referring to the payment terms can be added to the “Order” structure.

Priority	Source Type	Source No.	Source Structure Code	Source Name	Serialized Name
7	Record As Object	3	PAYMENTTERMS	Payment Terms	paymentterms

Then set the relation between the two.



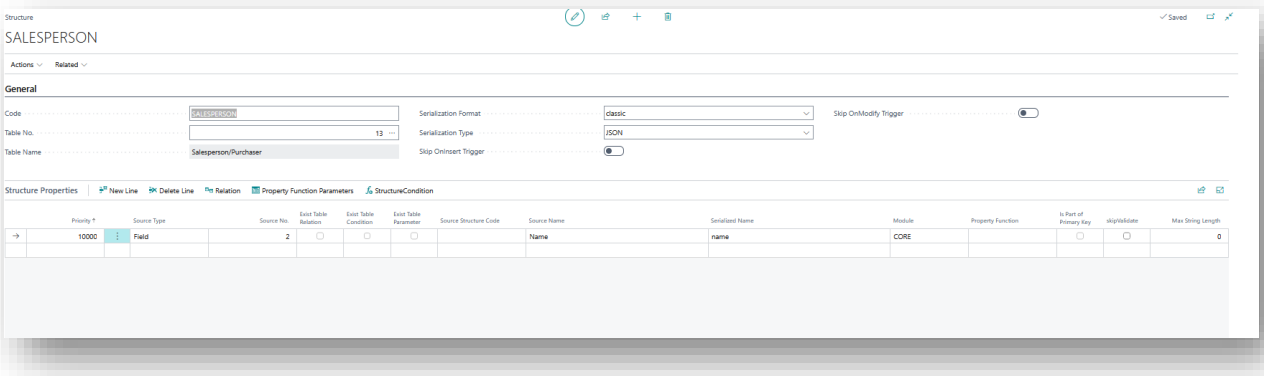
Now the serialization produces an output with a single JSON object as value of the property.

```
{
   "documenttype": 1,
   "selltocustomerno": "10000",
   "no": "101001",
   "billtocustomerno": "10000",
   "billtoname": "Adatum Corporation",
   "billtoname2": "",
   "billtoaddress": "Station Road, 21",
   "billtoaddress2": "",
   "billtocity": "Cambridge",
   "billtocontact": "Robert Townes",
   "yourreference": "OPEN",
   "shiptocode": "",
   "shiptoname": "Adatum Corporation",
   "shiptoname2": "",
   "shiptoaddress": "Station Road, 21",
   "shiptoaddress2": "",
   "shiptocity": "Cambridge",
   "shiptocontact": "Robert Townes",
```

```
    "orderdate": "2026-01-31",
    "postingdate": "2026-01-31",
    "shipmentdate": "2026-10-01",
    "salesline": [
       {
          "documenttype": 1,
          "selltocustomerno": "10000",
          "documentno": "101001",
          "lineno": 10000,
          "type": 2,
          "no": "1996-S",
          "locationcode": "",
          "postinggroup": "RESALE",
          "shipmentdate": "2026-03-29",
          "description": "ATLANTA Whiteboard, base",
          "description2": "",
          "unitofmeasure": "Piece",
          "quantity": 12,
          "outstandingquantity": 12,
          "qtytoinvoice": 12,
          "qtytoship": 12,
          "unitprice": 906.7,
          "unitcost(lcy)": 707.2,
          "vat%": 25,
          "linediscount%": 0
       }
    ],
    "paymentterms": {
       "code": "1M(8D)",
       "duedatecalculation": "1M",
       "discountdatecalculation": "8D",
       "discount%": 2,
       "description": "1 Month/2% 8 days",
       "calcpmtdisconcrmemos": false,
       "lastmodifieddatetime": "2026-10-01T01:11:18.6930000Z"
    }
 }
```


Add another line to retrieve the name of the salesperson of the document.

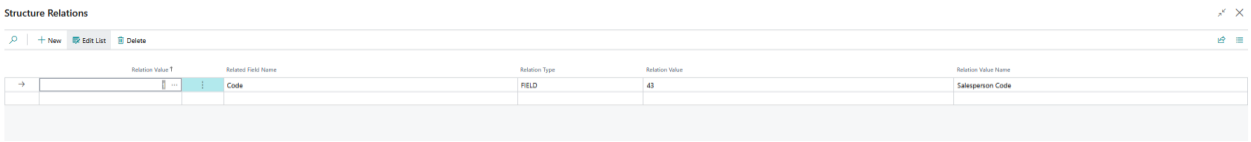
Create a structure linked to table 13 with a single property, the only one retrieved.



It is linked to the order.

Priority	Source Type	Source No.	Source Structure Code	Source Name	Serialized Name
8	Single Property As Value	13	SALESPERSON	Salesperson/Purchaser	salesperson/purchaser

The relation is set as in the previous cases.




```
{
   "documenttype": 1,
   "selltocustomerno": "10000",
   "no": "101001",
   "billtocustomerno": "10000",
   "billtoname": "Adatum Corporation",
   "billtoname2": "",
   "billtoaddress": "Station Road, 21",
   "billtoaddress2": "",
   "billtocity": "Cambridge",
   "billtocontact": "Robert Townes",
   "yourreference": "OPEN",
   "shiptocode": "",
   "shiptoname": "Adatum Corporation",
   "shiptoname2": "",
   "shiptoaddress": "Station Road, 21",
   "shiptoaddress2": "",
   "shiptocity": "Cambridge",
   "shiptocontact": "Robert Townes",
   "orderdate": "2026-01-31",
   "postingdate": "2026-01-31",
   "shipmentdate": "2026-10-01",
   "salesline": [
      {
         "documenttype": 1,
         "selltocustomerno": "10000",
         "documentno": "101001",
         "lineno": 10000,
         "type": 2,
         "no": "1996-S",
```

```
            "locationcode": "",
            "postinggroup": "RESALE",
            "shipmentdate": "2026-03-29",
            "description": "ATLANTA Whiteboard, base",
            "description2": "",
            "unitofmeasure": "Piece",
            "quantity": 12,
            "outstandingquantity": 12,
            "qtytoinvoice": 12,
            "qtytoship": 12,
            "unitprice": 906.7,
            "unitcost(lcy)": 707.2,
            "vat%": 25,
            "linediscount%": 0
         }
      ],
      "paymentterms": {
         "code": "1M(8D)",
         "duedatecalculation": "1M",
         "discountdatecalculation": "8D",
         "discount%": 2,
         "description": "1 Month/2% 8 days",
         "calcpmtdisconcrmemos": false,
         "lastmodifieddatetime": "2026-10-01T01:11:18.6930000Z"
      },
      "salesperson/purchaser": "Jim Olive"
   }
```


The JSON contains the whole structure of the order.

FUNCTIONS

Functions are provided by the modules to process the data further during serialization and deserialization.

FIXED

Sets a constant value exported for the property during serialization. During deserialization it replaces a missing property or overwrites its value.

Parameters

VALUE: the constant value to use.

FORMAT

During serialization it formats the value with the standard Business Central FORMAT specifications, as in https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-format-property

Parameters

FORMATPROPERTY: the format used to transform the value (e.g. <Year4>-<Month,2>-<Day,2>)

TRANSFORMATION

Transcodes the value in input and output. A translation model is defined in the setup and can be applied to several fields or tables.

See section TRANSFORMATION for details.

Parameters

MODEL: the transformation model used in serialization and deserialization.

MODULES

This version of the app includes the following modules. Each module is a set of features with its own commands. Each command behaves differently depending on the input parameters passed to it.

Each log is therefore a single execution of a command of a module.

CORE

The main module of the app (shown as BASE in the Italian user interface). It contains the serialization and deserialization features.

Commands

SERIALIZE

REQUIRED PARAMETERS

STRUCTURE

the code of the structure used to generate the serialized data. It must exist in the Structures table; the output is XML or JSON according to its configuration.

LINKEDRECORDID

The record the data is serialized from. Its table must match the table of the structure passed as parameter.

OPTIONAL PARAMETERS

SKIP and TOP

Only for SERIALIZE calls received through the API on JSON structures: SKIP is the number of records to skip and TOP the maximum number of records to return. If TOP is not passed, all the records after the skipped ones are returned. If SKIP is negative or greater than the number of records, the call fails with "Invalid Skip Parameter". The total number of records matching the filter is returned in the COUNT output parameter. See the example in section API CALL EXAMPLES.

SERIALIZEALL

REQUIRED PARAMETERS

STRUCTURE

the code of the structure used to generate the serialized data. It must exist in the Structures table; the output is XML or JSON according to its configuration.

DESERIALIZE

REQUIRED PARAMETERS

STRUCTURE

the code of the structure used to interpret the received data. It must exist in the Structures table; the input is read as XML or JSON according to its configuration.

INPUTDATA

Must contain the data to deserialize.

REPORT

A module that generates reports in Word, PDF or Excel format. It can be called through the API (the report is returned in base64 as output) or internally (the result is stored in the output data of the log).

Commands

PDF

Generates the report in PDF format.

REQUIRED PARAMETERS

REPORTNO

The number of the report to run.

LINKEDRECORDID

If the log has a linked record, the report is filtered on that record.

INPUTDATA

Can contain the request page parameters of the report.

E.g.:

```
<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement"
id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-
30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field
name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1)
SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1)
SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>
```

WORD

Generates the report in Word format, if the report supports it.

REPORTNO

The number of the report to run.

LINKEDRECORDID

If the log has a linked record, the report is filtered on that record.

INPUTDATA

Can contain the request page parameters of the report.

E.g.:

```
<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement"
id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-
30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field
name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1)
SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1)
SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>
```

EXCEL

Generates the report in Excel format, if the report supports it.

REPORTNO

The number of the report to run.

LINKEDRECORDID

If the log has a linked record, the report is filtered on that record.

INPUTDATA

Can contain the request page parameters of the report.

E.g.:

```
<?xml version="1.0" standalone="yes"?><ReportParameters name="VAT Statement"
id="12"><Options><Field name="StartDate">2023-07-05</Field><Field name="EndDateReq">2023-07-
30</Field><Field name="Selection">0</Field><Field name="PeriodSelection">1</Field><Field
name="VATPeriod" /></Options><DataItems><DataItem name="VAT Statement Name">VERSION(1)
SORTING(Field1,Field2)</DataItem><DataItem name="VAT Statement Line">VERSION(1)
SORTING(Field1,Field2,Field3) WHERE(Field4=1(46))</DataItem></DataItems></ReportParameters>
```

REST

This module makes REST calls to external services based on the log parameters.

Commands

SEND

REQUIRED PARAMETERS

URL

The URL of the remote service.

METHOD

The HTTP method used to call the remote service.

CONTENT-TYPE

The content type of the data sent, required only if the method has a message body.

E.g.: application/json, application/xml, application/octet-stream

OPTIONAL PARAMETERS

Optional parameters with subtype HEADER are added to the request headers.

INPUTDATA

The body of the message, if the HTTP method requires it. It must match the CONTENT-TYPE parameter.

Authentication

Authentication is set on the transceiver, not on the single log. The REST authentication data for an external service is entered in the Info parameters, so that it does not affect the log parameters.

MANDATORY PARAMETERS

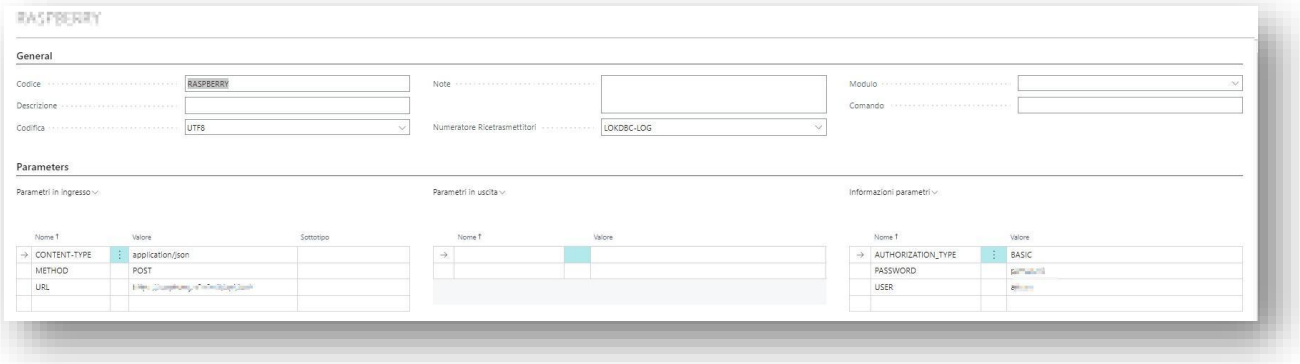
AUTHORIZATION_TYPE

The system supports BASIC and OAUTH2 authentication, besides anonymous calls.

BASIC

BASIC authentication requires two parameters:

USER and PASSWORD



OAUTH2

OAUTH2 authentication requires all the parameters needed to obtain a token:

CLIENTID

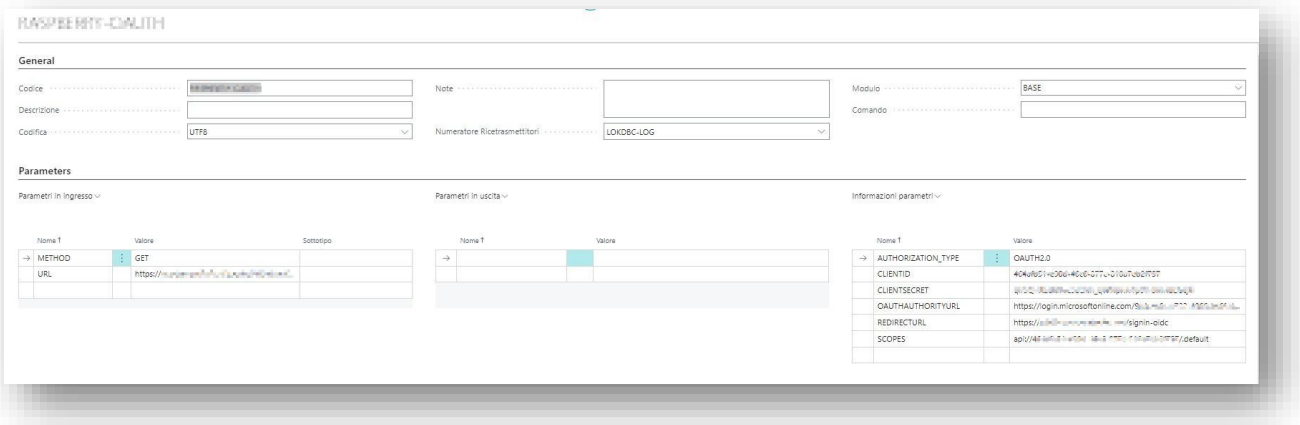
CLIENTSECRET

OAUTHAUTORITYURL

REDIRECTURL

SCOPES

The parameters follow the OAUTH2 standard; how to fill them in is outside the scope of this manual.



TRANSFORMATION

Introduction

A modular transcoding system that changes a value before serialization or deserialization without being tied to a single field.

Several structures often need the same translation of values: the goal is to avoid repeating the same list of pairs every time.

For example, a country must be transcoded in customers but also in sales and purchase document headers and in shipments, which are all different tables.



Modello	Tipo Operazione	Tipo di trasformazio...	Da	A
COUNTRY	Ricevi	Semplice	Argentina	AG
COUNTRY	Ricevi	Semplice	Regno unito	UK
COUNTRY	Ricevi	Semplice	Italia	IT
COUNTRY	Invia	Semplice	IT	Italia
COUNTRY	Invia	Semplice	UK	Regno unito
COUNTRY	Invia	Semplice	AG	Argentina

By linking the transformation function to a model, the model can be used wherever the same kind of value appears.

Currently the only transformation type is SIMPLE: a one-to-one replacement between an input and an output value, separate for each direction of the flow. The transformation type is extensible by other apps.

Fields

Model Code

The model, that is the set of transformations applied to the structure property.

Log Type

Defines the direction the line applies to: Send in serialization (data leaving Business Central), Receive in deserialization (data entering Business Central).

Transformation Type

Currently only SIMPLE: a one-to-one replacement of the From value with the To value.

From

The original value.

To

The translated value.

If the model has no line for the value to transform, the behavior depends on the "Enable Missing Transformation Error" field of the Setup.

API

The whole system has a single API endpoint, which unifies and simplifies every communication received from outside.

http://[URL]/[SERVICE]/api/lookie/dataBridge/v2.0/companies([IDCOMPANY])/logs?tenant=[TENANT]

The variable parts are the URL, the TENANT, the SERVICE name and the ID of the company the calls are sent to.

All calls to this address must be POST requests with an application/json body structured as follows.

```
{
   "transmitterCode": "",
   "module": "",
   "command": "",
   "input": "",
   "parameters": [
      {
         "name": "",
         "value": ""
      }
   ]
}
```

transmitterCode

the code of the sender of the call. It must exist in the Transceivers table, otherwise the call is rejected.

module

the module that provides the command to run. It is not needed if no modules other than the base one are installed.

command

the command to run within the requested module.

input

the actual content of the call: any kind of data, from XML to JSON to a base64 string. The code of the command processes it accordingly.

parameters

Parameters of the requested command can (and sometimes must) be added to these main properties.

The parameters are passed as name/value pairs.

API CALL EXAMPLES

SERIALIZE

```
{
   "transmitterCode": "VSCODE",
   "command":"SERIALIZE",
   "input":"" ,
   "parameters":[
      {
         "name":"STRUCTURE",
         "value":"ITEM"
   } ]
}
```

This call runs a SERIALIZE command for the sender VSCODE. The command requires a STRUCTURE parameter with the structure used to extract the data.

The result is

```
{
   "@odata.context":
   "http://[URL]:7048/[SERVICE]/api/lookie/dataBridge/v2.0/$metadata#companies([IDCOMPANY])/l
   ogs/$entity",
   "@odata.etag": "W/\"JzIwOzEzODUyMjgxMjI2MjA2NjI1NzYzMTswMDsn\"",
   "no": "LOG-0000011868",
   "transmitterCode": "VSCODE",
   "command": "SERIALIZE",
   "module": "BASE",
   "input": "",
   "output":
   "[{\"ProductID\":\"1\",\"ProductName\":\"Chai\",\"genprodpostinggroup\":\"MISC\",\"vatprod
   postinggroup\":\"VAT20\",\"inventorypostinggroup\":\"RAW
   MAT\"},{\"ProductID\":\"10\",\"ProductName\":\"Ikura\",\"genprodpostinggroup\":\"MISC\",\"
   vatprodpostinggroup\":\"VAT20\",\"inventorypostinggroup\":\"RAW
   MAT\"},{\"ProductID\":\"1000\",\"ProductName\":\"Bicycle\",\"genprodpostinggroup\":\"RETAI
   L\",\"vatprodpostinggroup\":\"VAT20\",\"inventorypostinggroup\":\"FINISHED
   GOODS\"},{\"ProductID\":\"1000B\",\"ProductName\":\"Bicycle\",\"genprodpostinggroup\":\"MI
   SC\",\"vatprodpostinggroup\":\"VAT20\",\"inventorypostinggroup\":\"RAW MAT\"}]"
}
```

An output property with the result of the command is added to the call data.

For the SERIALIZE command the input property can contain the filters to set on the table.

```
{
   "transmitterCode": "VSCODE",
   "command":"SERIALIZE",
   "input":"WHERE(No.=FILTER('LS*'))" ,
   "parameters":[
      {
         "name":"STRUCTURE",
         "value":"ITEM"
   } ]
}
```

In this example the table of the ITEM structure is filtered with LS* on the “No.” field.

For JSON structures the result can be returned in pages with the SKIP and TOP parameters.

```
{
   "transmitterCode": "VSCODE",
   "command":"SERIALIZE",
   "input":"" ,
   "parameters":[
      {
         "name":"STRUCTURE",
         "value":"ITEM"
      } ,
      {
         "name":"SKIP",
         "value":"10"
      } ,
      {
         "name":"TOP",
         "value":"5"
   } ]
}
```

In this example the first 10 records of the table of the ITEM structure are skipped and at most the next 5 are returned. The COUNT output parameter contains the total number of records matching the filter: increase SKIP to get the next pages.

DESERIALIZE

Back to the deserialization example of section Structure examples.

The JSON of that example is passed as follows.

```
{
   "transmitterCode": "VSCODE",
   "command":"DESERIALIZE",
   "input":"{
      \"ProductID\": \"1000C\",
      \"ProductName\": \"Bicycle\",
      \"genprodpostinggroup\": \"MISC\",
      \"vatprodpostinggroup\": \"VAT20\",
      \"inventorypostinggroup\": \"RAW MAT\"
      }
      " ,
   "parameters":[
      {
         "name":"STRUCTURE",
         "value":"ITEM"
   } ]
}
```

The call inserts a new record in the table of the ITEM structure.

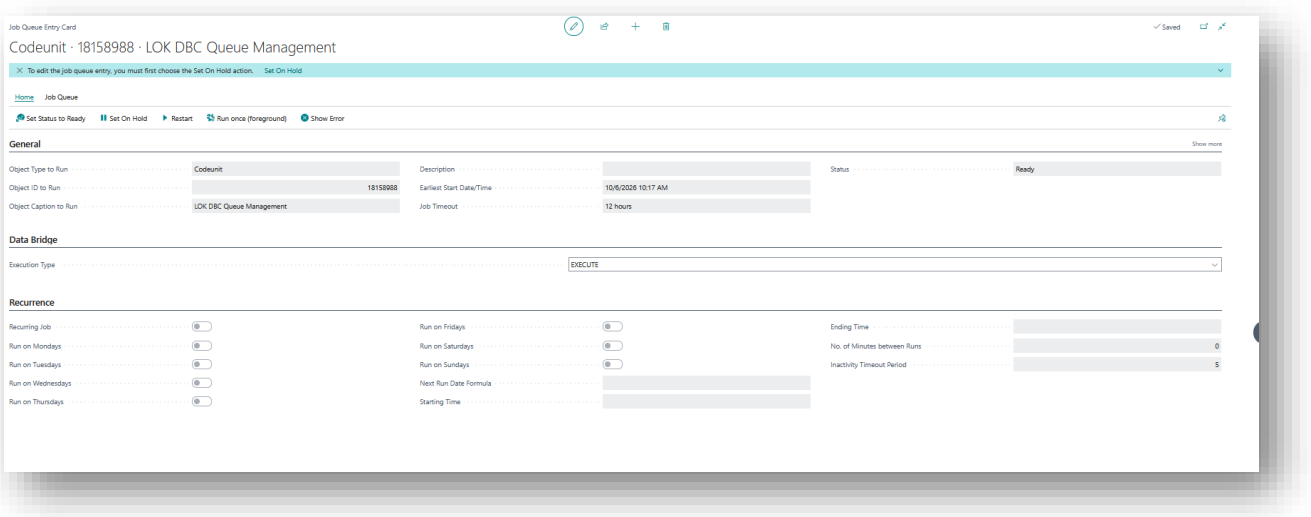
Job queue

A job queue can be used to execute the logs received through the API.

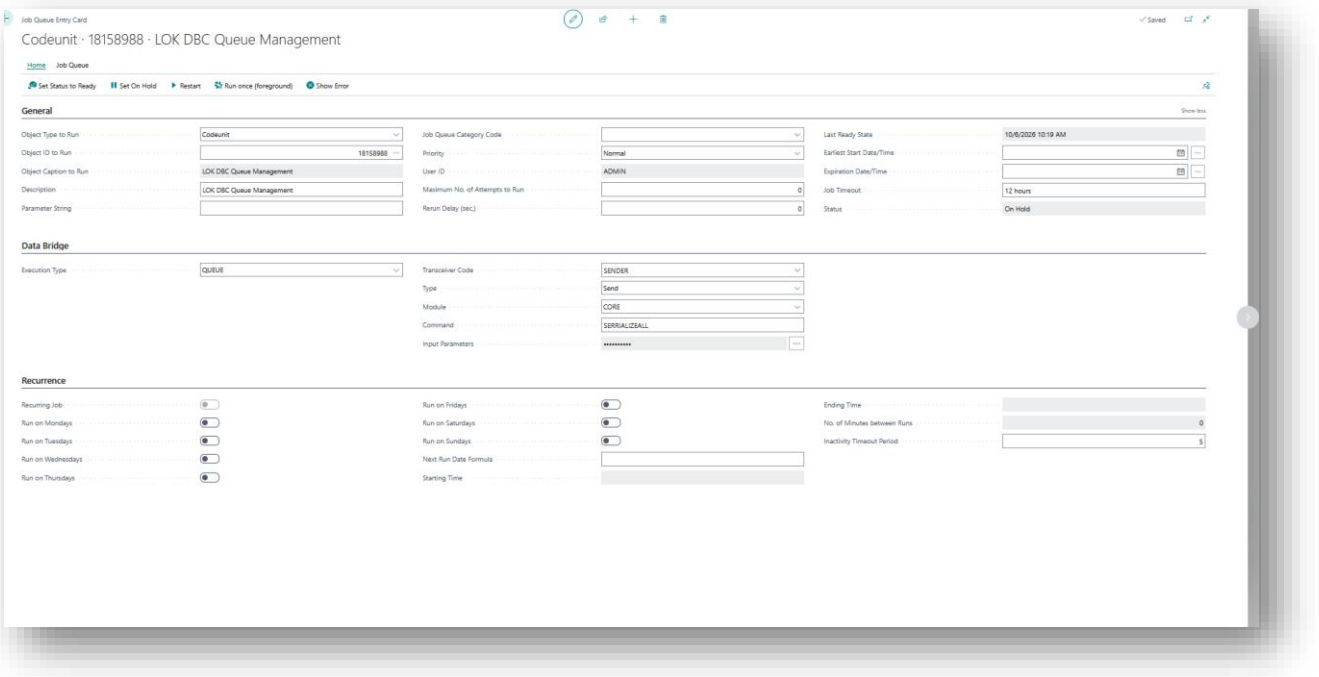
For a line received through the API to stay “Queued” instead of being processed immediately, the API call must include the “ASYNC” parameter.

```
{
   "transmitterCode": "VSCODE",
   "command":"DESERIALIZE",
   "input":"{
      \"ProductID\": \"\",
      \"ProductName\": \"Bicycle\",
      \"genprodpostinggroup\": \"MISC\",
      \"vatprodpostinggroup\": \"VAT20\",
      \"inventorypostinggroup\": \"RAW MAT\"
      }
      " ,
   "parameters":[
      {
         "name":"STRUCTURE",
         "value":"ITEM"
      } ,
      {
         "name":"ASYNC"
   } ]
}
```

With the ASYNC parameter (no value needed) the line is inserted without being processed in real time. The sender only receives the confirmation of receipt, without an output. The job queue can be scheduled with a minimum recurrence of one minute.



The job queue can also create a new log line on a schedule, which is then processed by the standard flow.



In this case a SERIALIZEALL line is created and the structure to use is set in the input parameters.

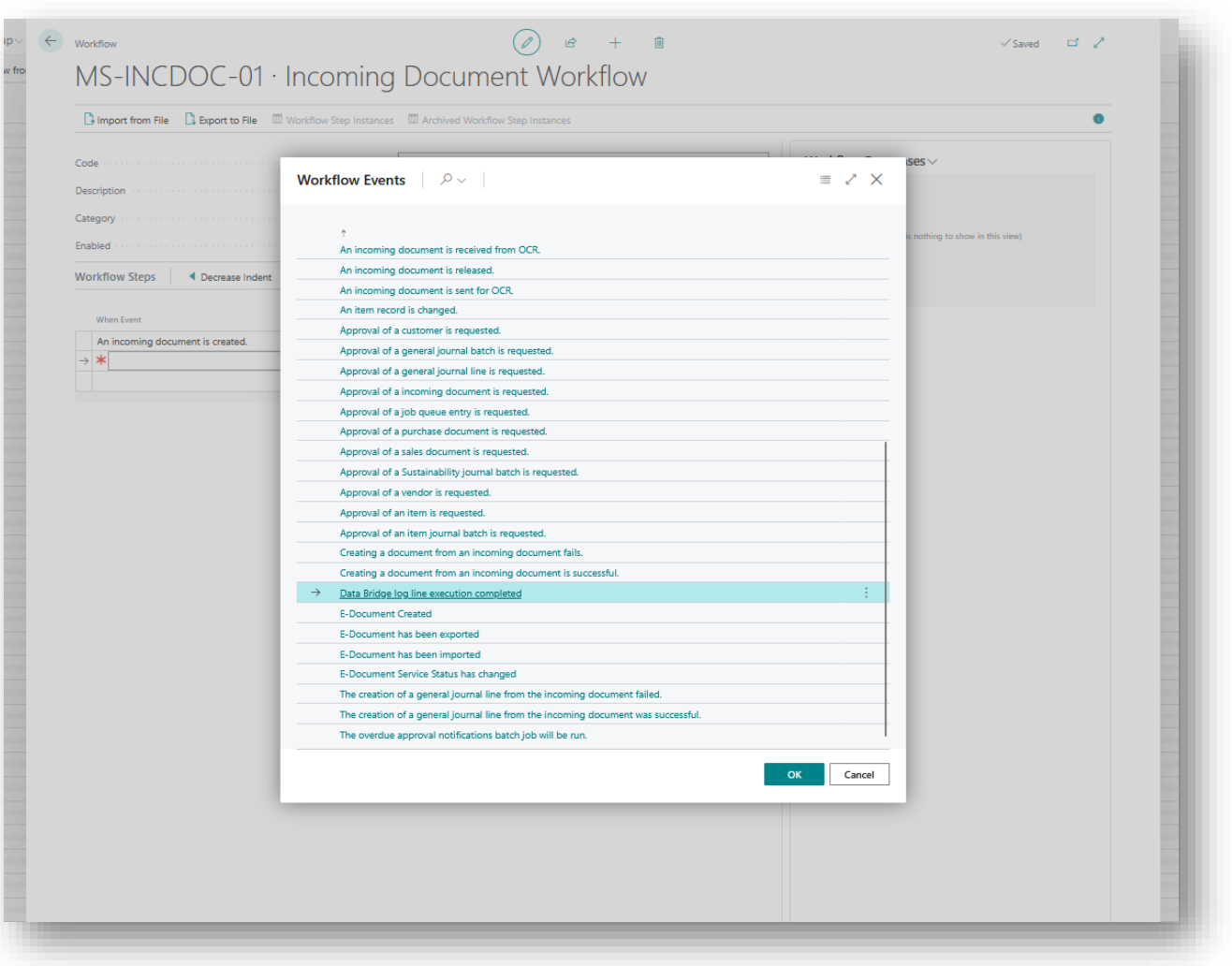
For example, all the records of a table can be sent automatically to a remote service at regular intervals.

Another example is printing a summary report and uploading it to an external document management system.

Workflow

The logs are integrated with the standard Business Central workflows through an event, a condition and a response.

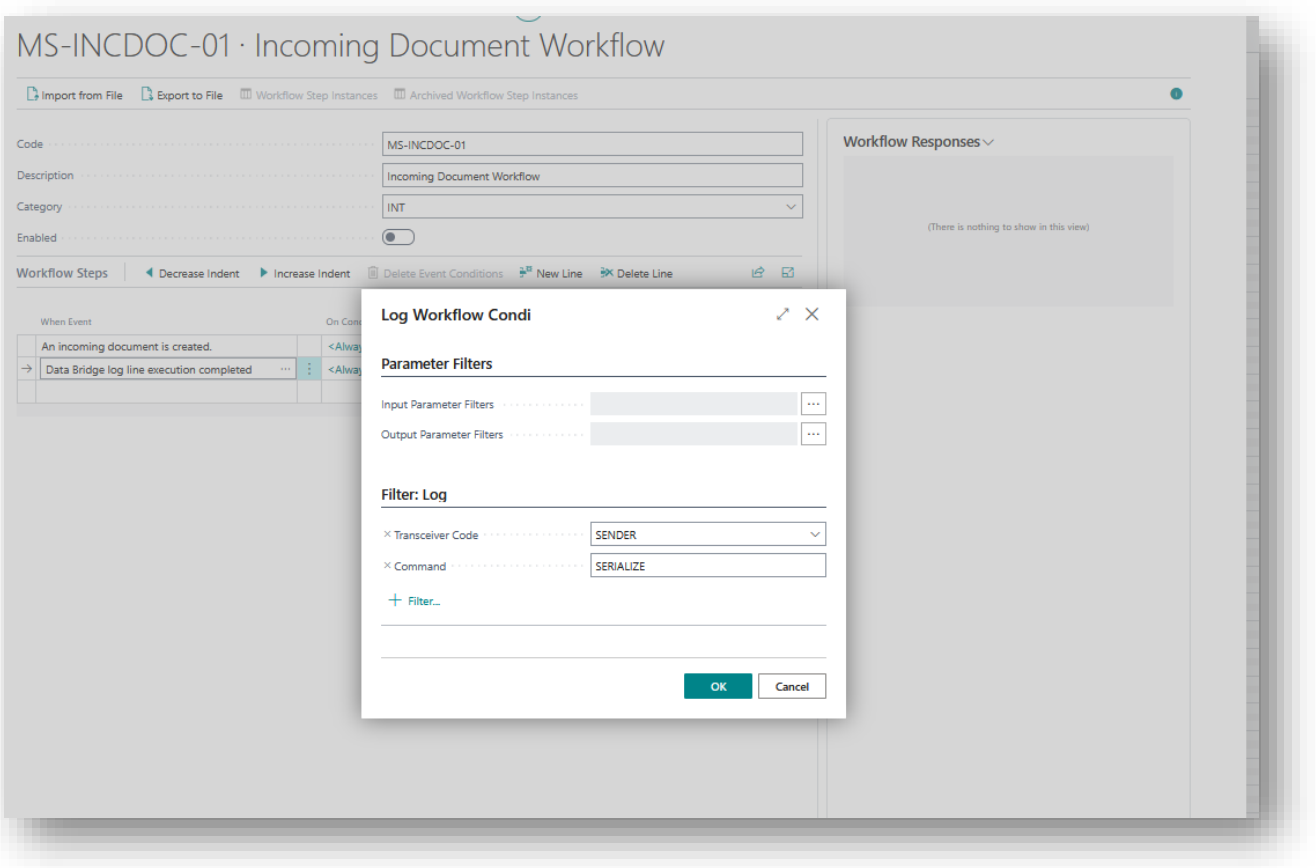
EVENT



The event fires when a log line completes successfully and its status becomes Processed.

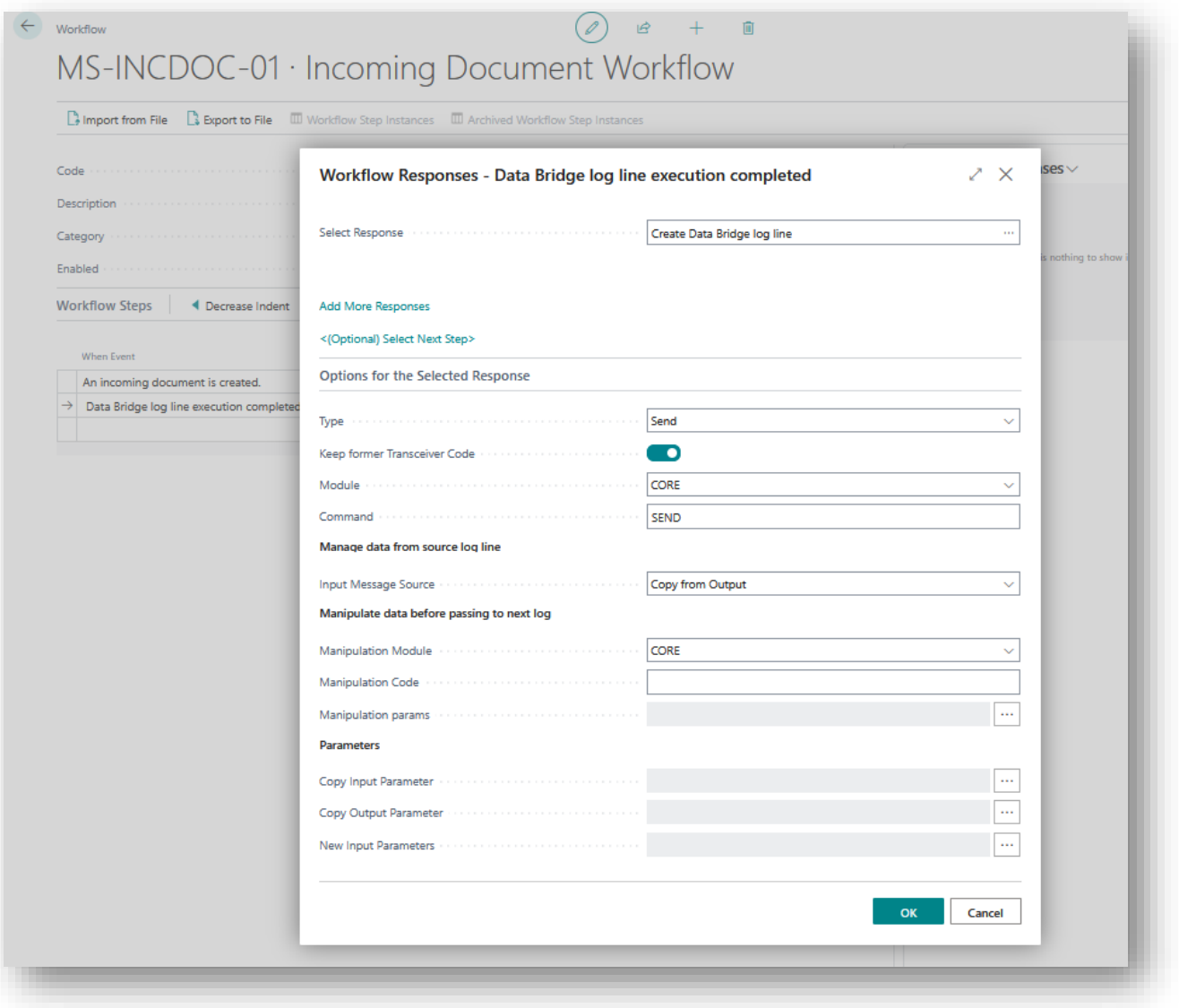
A chained event can therefore be added.

CONDITION



When the event is the completion of a log line, the condition allows, besides the standard record filters, to filter on the input and output parameters with name/value pairs.

RESPONSE



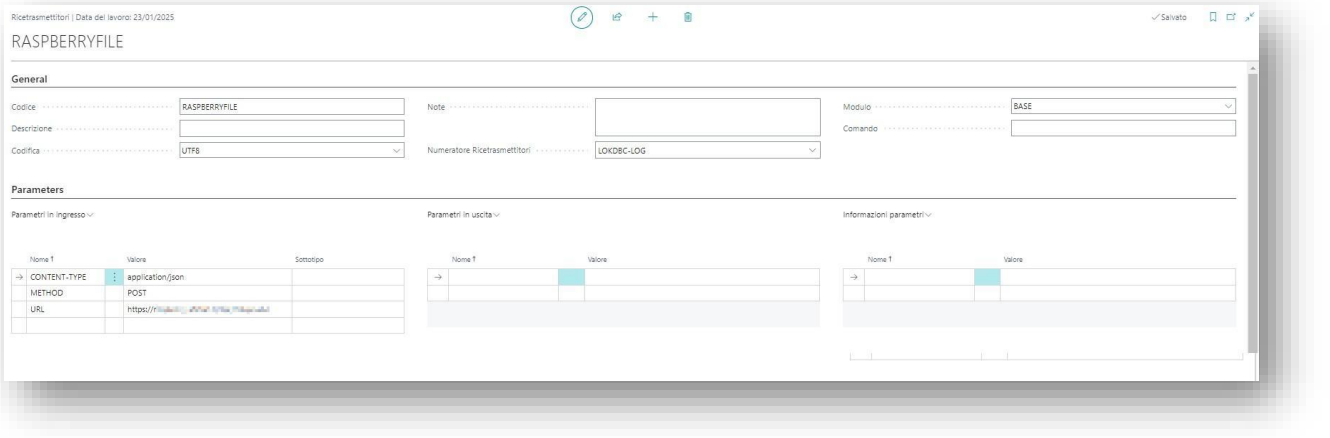
A log line can be created as the response to any workflow event. Input parameters can be set and, if the event comes from a log, the output data and parameters of the first log can be passed as input to the new one.

Modules and commands added by extensions are fully supported here, so workflow responses can run complex code without extending the standard workflow codeunits.

USAGE EXAMPLE

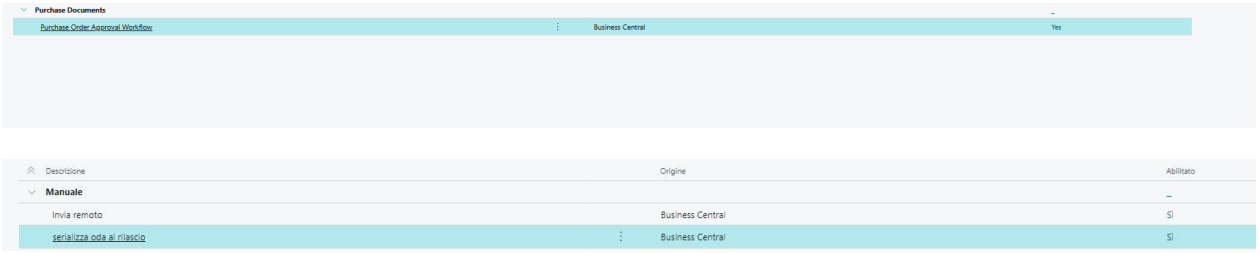
In this example a JSON with the data of a purchase order is sent to a remote REST service when the order is released.

First, define the destination of the call.

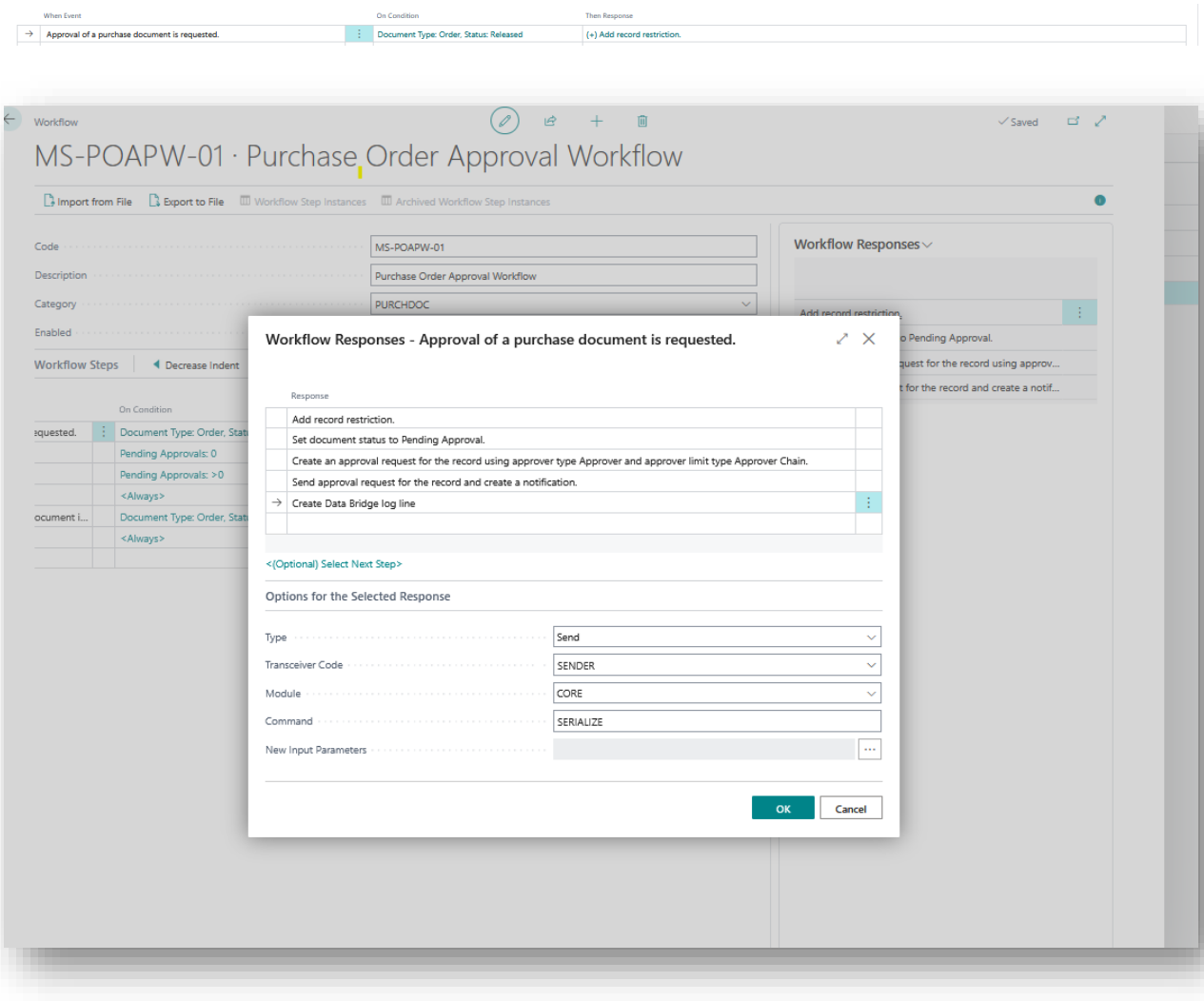


Set a URL and a method (POST in this case).

In the workflows, create the following logic:



Every time a purchase order is released, a new Data Bridge log line is created.



The line simply serializes the released document.

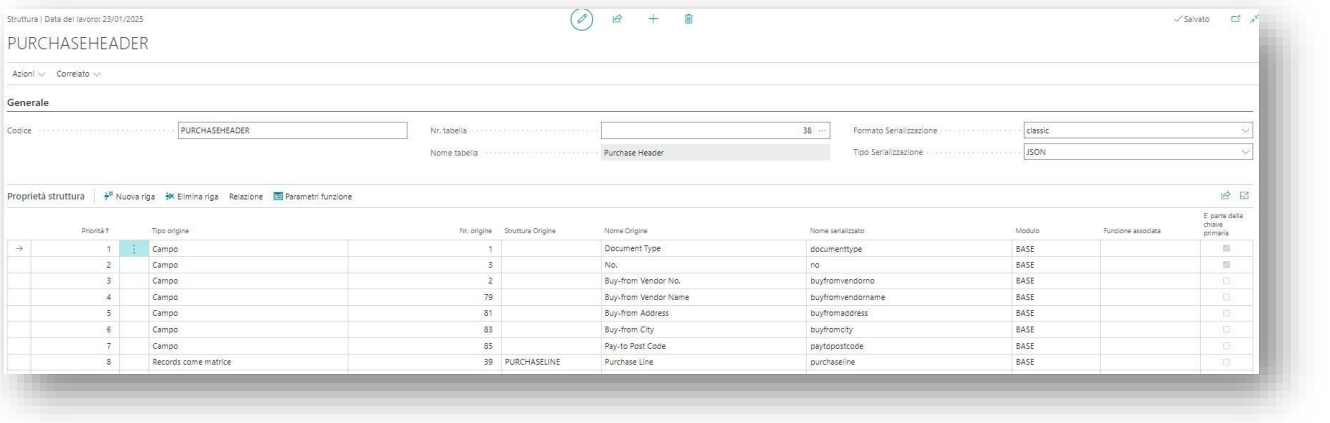
With the assist button next to “New Input Parameters” the parameter needed for the serialization is defined:



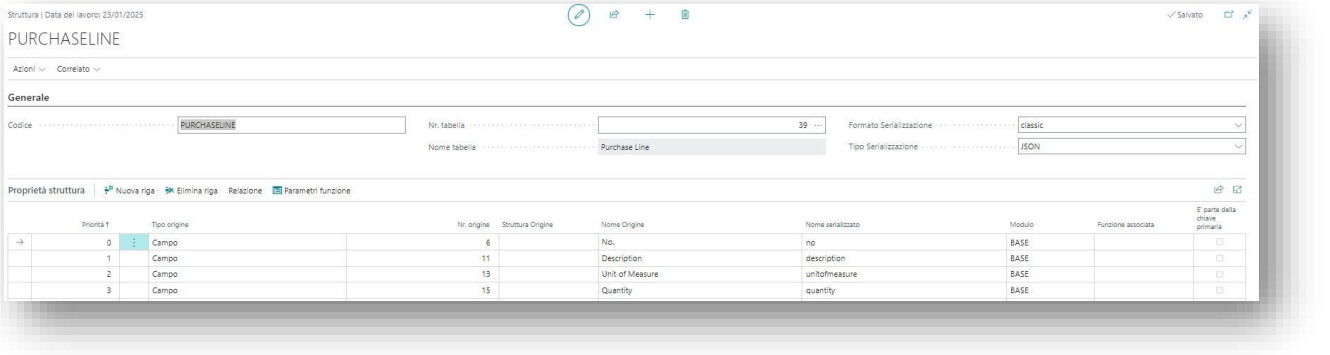
the structure to use, “STRUCTURE”, with the value “PURCHASEHEADER”.

Additional parameters can be added here, to be passed to the next workflow steps or used as conditions.

Then create a structure with the same code that retrieves the data from tables 38 and 39.



And the line details.



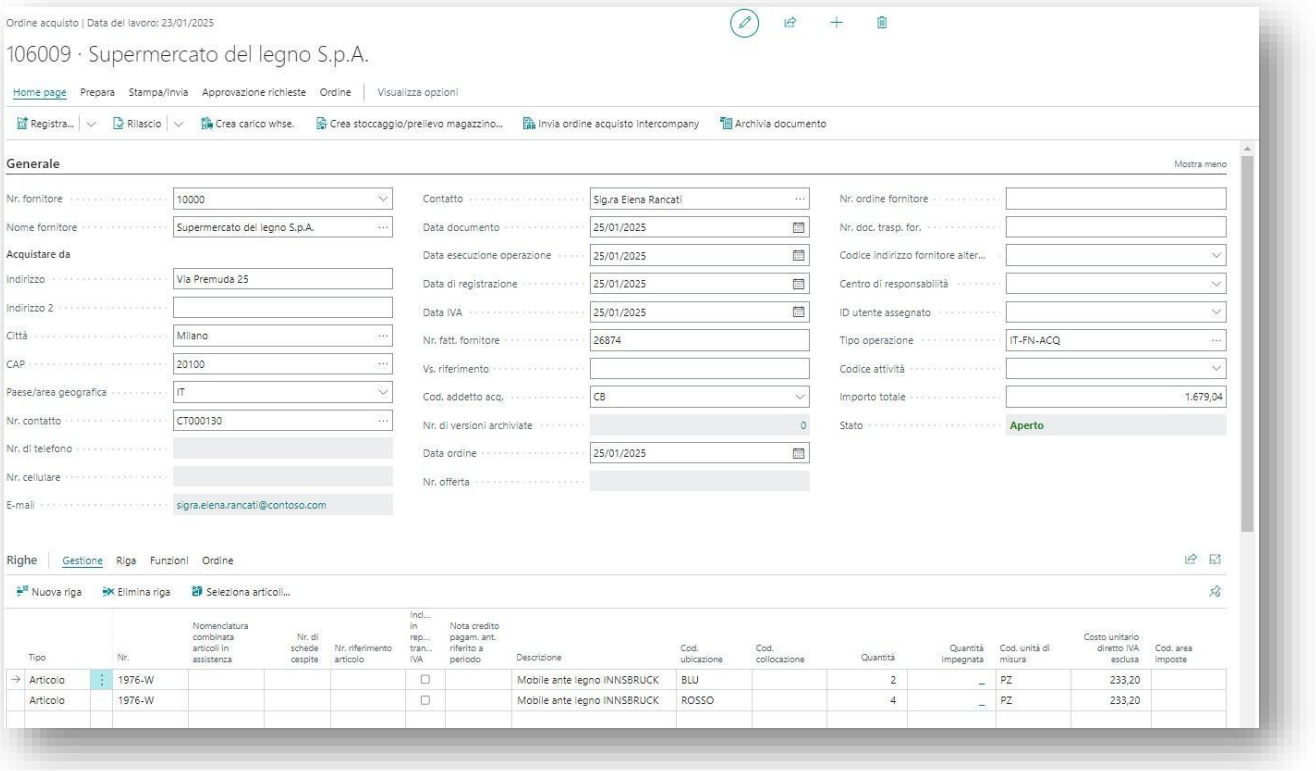
This configuration generates a JSON like the following.

```
{
   "documenttype": 1,
   "no": "104004",
   "buyfromvendorno": "40000",
   "buyfromvendorname": "Sabelli && Fioretti S.r.l.",
   "buyfromaddress": "Corso Colombo 13",
   "buyfromcity": "Reggio Emilia",
   "paytopostcode": "42100",
   "purchaseline": [
      {
         "no": "LS-10PC",
         "description": "Altoparl., bianco per PC",
         "unitofmeasure": "Scatola",
         "quantity": 40
      },
      {
         "no": "LS-150",
         "description": "Altop., ciliegia, 150W",
         "unitofmeasure": "Piece",
         "quantity": 20
      }
] }
```

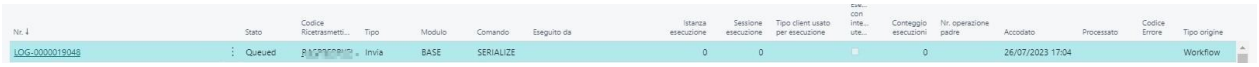
A preview is available with the Serialize action of the structure card.

Now, without writing code, release any purchase order.

Take the order

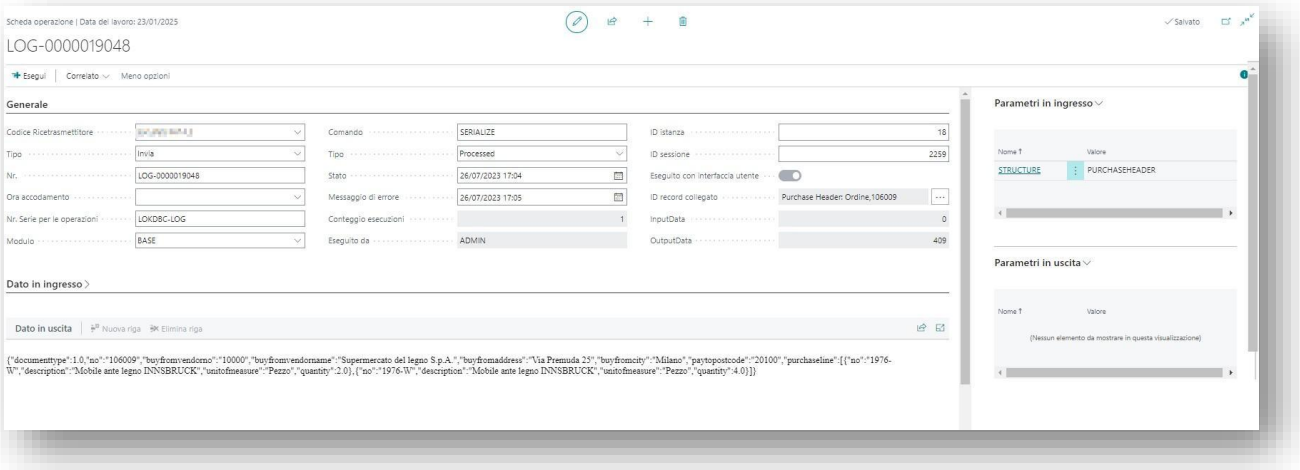


The release creates the following log



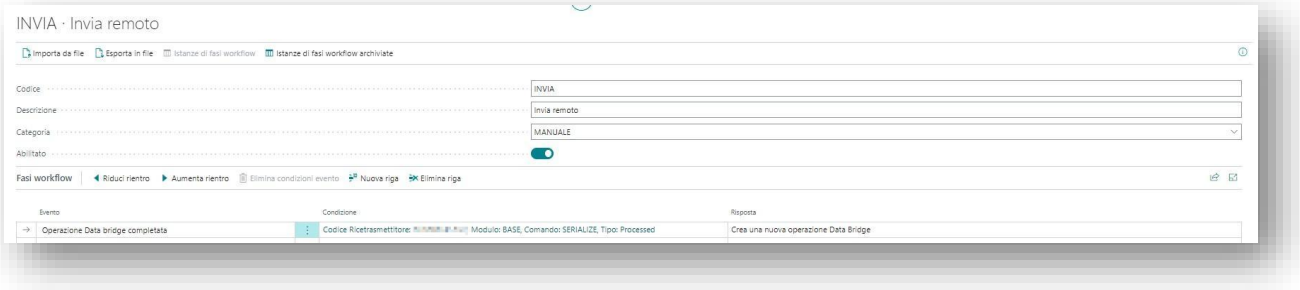
The line comes from a workflow, not from an API call; the card shows additional details.

The record is created as Queued and is executed by the scheduled job queue.



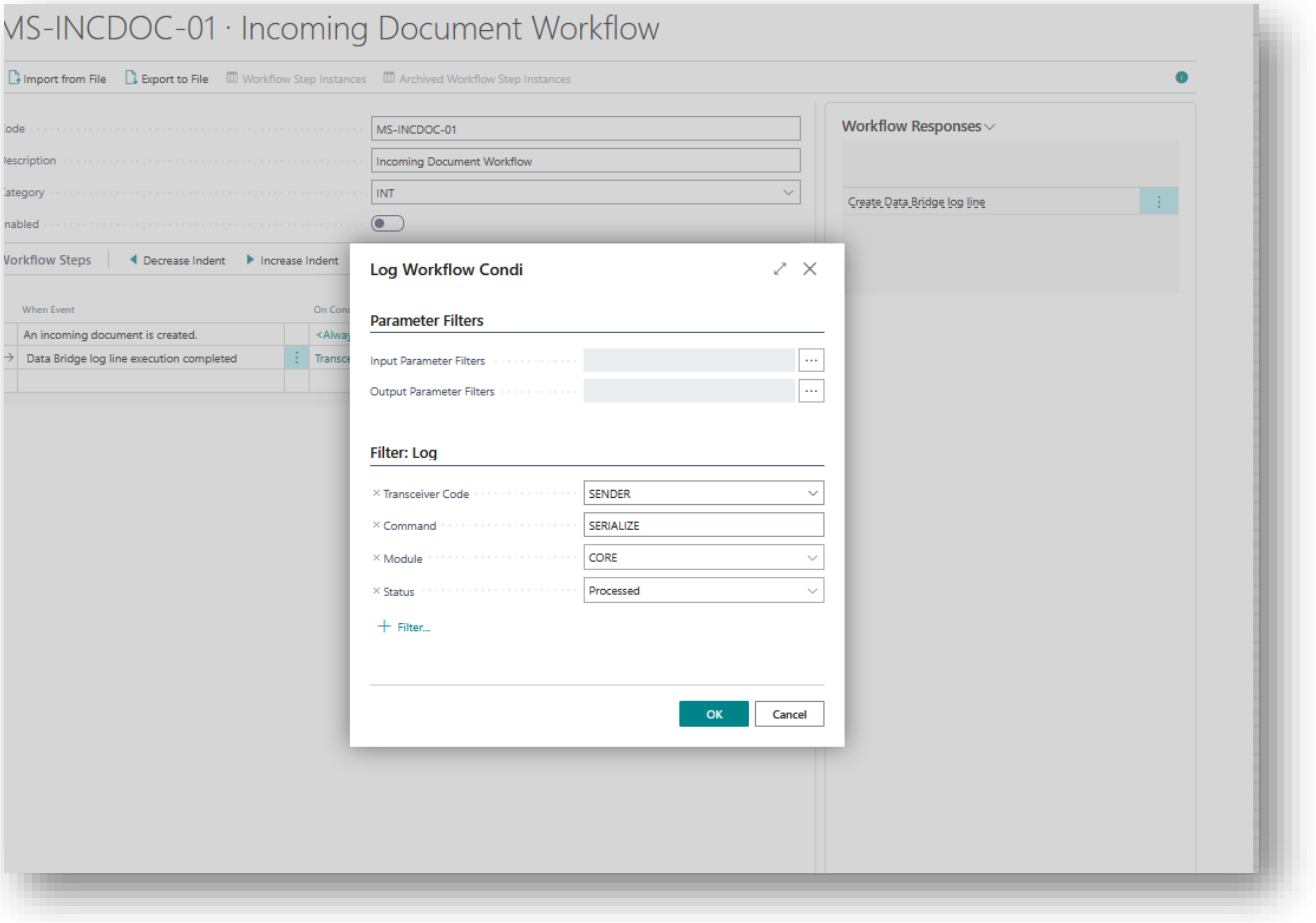
The STRUCTURE parameter used for the serialization is “PURCHASEHEADER” and the output data contains the result of the serialization: the JSON. The linked record ID gives quick access to the record card, and the card also shows the size of the data.

Back in the workflow, add another step to the flow.



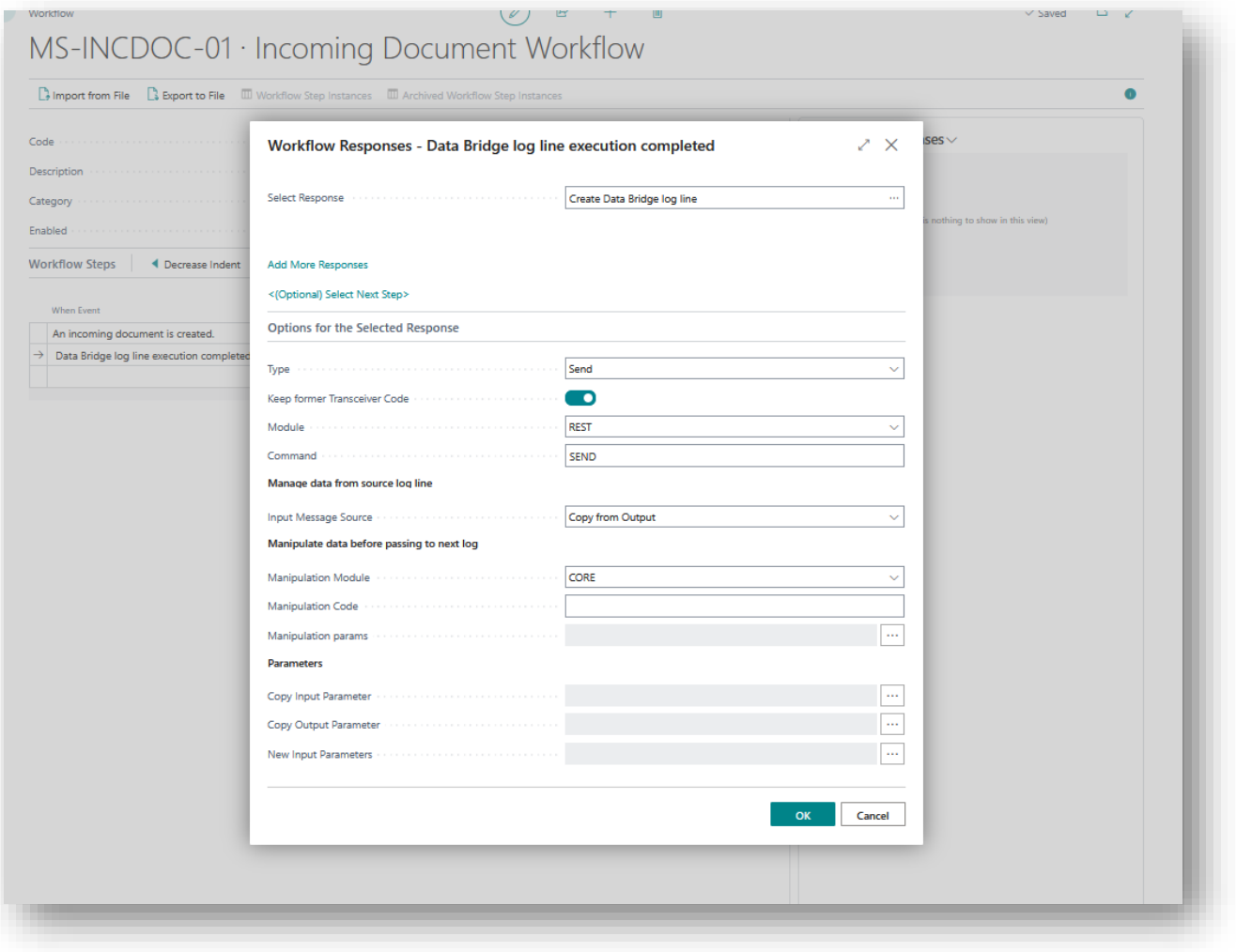
Every time a line is executed the event fires.

The conditions are not limited to the table data but also include the parameters.



For example, the flow can start only when the record comes from a serialization and the STRUCTURE parameter has a given value.

This allows different behaviors for different serializations.

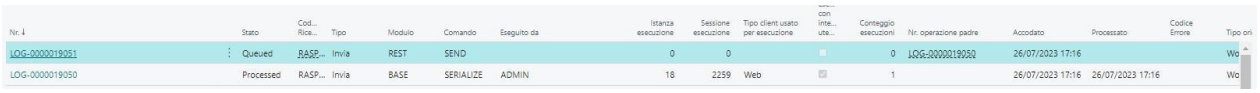


As response, if the conditions are met, a new Data Bridge line is created that uses the output data of the first line as input and makes a REST SEND.

After enabling the workflow, reopen and release the same purchase order



which, once executed, generates



The two lines are linked by the “Parent No.” field. The execution sends the data to the remote destination.



All this without writing a single line of code.

In the same way a report could be sent by chaining a REPORT log with a REST SEND log.

Besides additional extensions that implement new communication channels (SharePoint, file storage, blob storage, mail), the system provides events and interfaces to extend it with code.



Il tuo partner per Microsoft Dynamics 365 Business Central

Chi siamo

LOOKie è la business unit del Gruppo Omega dedicata al mondo Microsoft Dynamics 365 Business Central, per rispondere alle esigenze di aziende con processi complessi e sfidanti.

Un team di esperti in Business Central al tuo fianco per sviluppare soluzioni su misura e innovative.

Il gruppo OMEGA

Dal 1998 Omega affianca le imprese nell'innovazione dei processi operativi e nel miglioramento tecnologico.

Grazie alle esperienze maturate in ambito di software gestionali e ERP, sosteniamo le imprese lungo tutto il percorso di digital transformation offrendo soluzioni e servizi ad alto contenuto innovativo, per supportarne la crescita e il conseguimento dell'eccellenza operativa con una visione improntata al miglioramento tecnologico continuo

I numeri del gruppo

6 **SEDI** OPERATIVE

24,5 **MLN €** FATTURATO GRUPPO 2025

157 **TEAM** PROFESSIONISTI

1056 **PROGETTI** DI DIGITALIZZAZIONE

CONTATTI

+39 0422.823810 MARKETING@OMEGAGRUPPO.IT WWW.LOOKIE.IT