

## Manuale Utente e Ambiti di Utilizzo per ntJobApp

### 1. Introduzione e Scopo

Una ntJobApp è un'applicazione progettata per funzionare in **modalità batch**. L'architettura utilizza la classe ncJobApp e l'istanza jData per orchestrare queste micro applicazioni batch.

Lo scopo di una ntJobApp è triplice:

1. Leggere un file .ini passato come parametro.
2. Eseguire uno o più comandi richiesti dal chiamante.
3. Restituire un file .ini di risultato delle elaborazioni. Il file di output .ini verrà rinominato con estensione .end.

### 2. Logica di Funzionamento per l'Esecutore

L'esecuzione di una ntJobApp segue una logica sequenziale standard:

1. **Creazione Istanza:** Viene creata l'istanza di ncJobApp nella variabile globale jData.
2. **Avvio (Start()):** Viene eseguito self.Start(), che legge il file .ini passato come parametro (o lo crea) e lo salva in self.dictJobs.
  - o Se sResult (il risultato dell'operazione) è diverso da "", l'applicazione esce con errorlevel 1.
3. **Esecuzione (Run()):** Viene eseguito self.Run(), che esegue ogni singolo job definito in self.dictJobs in sequenza.
4. **Finalizzazione (End()):** Viene eseguito self.End(sResult).

### 3. Istruzioni per l'Esecuzione

Per eseguire una ntJobApp, è necessario avviare l'applicazione passando come **primo parametro** il percorso completo del file di configurazione .ini.

#### 3.1. Il File di Input (.ini)

Il file .ini deve contenere almeno la sezione [CONFIG] e una o più sezioni di *job*.

**Esempio di file ntjobsapp.ini letto all'avvio:**

Ini, TOML

[CONFIG]

TYPE=NTJOBS.2.0

EXIT=TRUE ; Se "TRUE", l'applicazione esce in caso di errore di un job.

LOG="FILE.LOG" ; Nome facoltativo del file di Log.

NAME=ID\_APPLICAZIONE

[JOB1\_ID]

ACTION=NOME\_AZIONE

FILE.ID1=PathFile1

PARAM.ID1=Valore

#### Validazioni cruciali in Start():

- **Sezione CONFIG:** Deve essere presente nel dizionario letto in self.dictJobs, altrimenti l'applicazione esce con l'errore "Sezione CONFIG non trovata".
- **Chiave COMMAND:** In ogni sezione job (eccetto CONFIG), deve esserci la chiave COMMAND (o ACTION nell'esempio), altrimenti l'errore viene accodato a sResult.
- **Verifica file:** Se ci sono chiavi che cominciano per FILE., l'applicazione verifica che il file esista. Se un file non esiste, l'errore viene accodato a sResult, ma l'esecuzione **non** viene interrotta in questa fase.

### 3.2. Il File di Output (.end)

Al termine dell'esecuzione, il metodo End(sResult) salva il contenuto di self.dictJobs come file .ini nel percorso self.sJobEnd. self.sJobEnd è il nome di self.sJobIni con l'estensione cambiata in .end.

Il file .end conterrà i risultati di ogni job, inclusi:

- RETURN.TYPE (es. E=Errore, ""=OK).
- RETURN.VALUE (Messaggio di ritorno facoltativo).
- TS.START e TS.END (Timestamp di inizio e fine).
- Chiavi con prefisso FILE.RETURN. (riferimenti a file generati dal job).

### 3.3. Codici di Risultato (Errorlevel)

Al termine dell'esecuzione, la ntJobApp restituisce un codice di risultato (errorlevel) che indica lo stato finale:

Codice	Descrizione	Significato
0	Tutto a posto	L'applicazione è terminata correttamente, tutti i job sono stati eseguiti.

Codice	Descrizione	Significato
1	Non è riuscito ad arrivare all'esecuzione dei job	Errore critico in fase di avvio (Start()), il file .ini di fine elaborazione non viene generato.
2	Uno o più job è terminato con un errore	Uno o più job all'interno del file .ini sono terminati con un errore.

#### 4. Ambiti di Utilizzo e Vantaggi vs. Automazione Web (Clic e Attese)

L'architettura ntJobApp è specificamente progettata per processi di automazione **batch** e offre significativi vantaggi rispetto a soluzioni che richiedono interazione grafica (come clic e attese su un sito web).

Ambiti Tipici di Utilizzo di ntJobApp	Vantaggi Chiave Rispetto all'Automazione Web (GUI)
<b>Processi Asincroni:</b> Elaborazioni complesse o massive, come la generazione di report notturni, che non devono bloccare l'utente.	<b>Efficienza e Velocità:</b> Essendo un processo di <i>backend</i> , non ha overhead grafici, non richiede l'attesa di rendering della pagina o l'interazione umana, riducendo drasticamente i tempi morti.
<b>Integrazione Dati:</b> Spostamento, conversione o elaborazione massiva di file (es. da FTP a un database), processi che lavorano "sotto il cofano".	<b>Robustezza e Affidabilità:</b> Non è soggetta a errori dovuti a cambiamenti nell'interfaccia utente (come un bottone che cambia ID) o problemi di timeout della rete dovuti al caricamento di una pagina.
<b>Orchestrazione Strutturata:</b> Chiamare diverse applicazioni sequenzialmente, dove i risultati e i file di ritorno di un'azione vengono gestiti come input per la successiva.	<b>Tracciabilità Standardizzata:</b> Il formato .ini in ingresso e l'output .end garantiscono un formato di status uniforme e leggibile dal chiamante, inclusi i file di ritorno generati.
<b>Schedulazione:</b> Esecuzione di task automatici a orari predefiniti (come Cron o Task Scheduler).	<b>Facilità di Automazione:</b> L'esecuzione da linea di comando con file di parametri la rende ideale per qualsiasi sistema di <i>job</i>

<b>Ambiti Tipici di Utilizzo di ntJobApp</b>	<b>Vantaggi Chiave Rispetto all'Automazione Web (GUI)</b>
	<i>scheduler</i> , richiedendo solo il percorso del file .ini.

---