

An aerial photograph of a historic Italian city, likely Florence, showing a dense cluster of buildings with terracotta roofs and a river winding through the center. A large bridge with multiple arches spans the river in the lower half of the image. The sky is a hazy, warm orange-brown color.

DELPHIDAY

italian conference

Delphi Blocks

Il package manager definitivo?



LUCA MINUTI



luca.minuti.it



luca.minuti@gmail.com



dev.to/lminuti



github.com/lminuti



www.linkedin.com/in/luca.minuti

DELPHIDAY

italian conference

9-10 Giugno 2026
Piacenza



wintech
italia

OPEN-SOURCE PROJECTS

github.com/lminuti

WiRL

github.com/delphi-blocks/WiRL

Delphi Blocks

github.com/delphi-blocks/blocks

MCPCConnect

github.com/delphi-blocks/MCPCConnect

DELPHIDAY

italian conference

9-10 Giugno 2026
Piacenza



wintech
italia



AGENDA

1. Perché Blocks?
2. Demo
3. Installare app con WinGet
4. I profili di Delphi
5. Il concetto di Workspace
6. Package e gestione delle dipendenze
7. Come funziona e come creare il Manifest



Introduzione

1



Perché BLOCKS

L'esigenza nasce dall'installazione di WiRL: complessa e con diverse dipendenze. Guardando le alternative disponibili, nessuna soddisfaceva pienamente:

GetIt — Integrato in Delphi ma **senza gestione delle dipendenze**, funziona solo a livello di progetto

Boss — Scritto in GO... (boss.json, lavora per progetto)

Tms smart setup — non aveva supporto per le versioni e dipendenze



PowerShell

```
irm https://wirl.delphiblocks.dev/install.ps1 | iex
```

- L'uso di PowerShell permette di evitare lo scaricamento manuale
- La base: **Invoke-RestMethod**, **Invoke-Expression**
- Provo a sviluppare un **prototipo** con l'aiuto della AI e... funziona!
- Ma non conosco PowerShell, ogni modifica minima mi richiede un sacco di tempo e sento di non avere il controllo



Rewrite in Delphi

- Un prototipo in pochissimo tempo mi ha dato un certa fiducia
- Anche solo la compilazione con cattura dell'output non è complicata mi ci avrei dovuto studiare API che non ricordo o aggiungere dipendenze ad librerie esterne
- Riparto da zero e in poco tempo ho una versione che fa le stesse cose ma in Delphi
- A quel punto parto col refactoring perché sono in grado di capire cosa funziona e cosa no

demo time





Usare blocks

2



WinGET

- L'utilizzo di **WinGET** è fondamentale perché l'idea era quella di avere meno passaggi possibili
- Non l'avevo mai usato molto ed è stato interessante studiarlo
- Il primo upload ha impiegato **un paio di giorni**
- I successivi **poche ore** (una volta rifiutato probabilmente per avevo compilato in release)
- Gestito da volontari



WinGET

1. Preparare il setup (Inno, NSIS, MSI/WiX e MSIX, ecc.)
2. Installare wingetcreate
`winget install Microsoft.WingetCreate`
3. Crea un Token su GitHub
4. Salva il Token
`wingetcreate token --store --token <TOKEN>`
5. Crea e invia il manifest
`wingetcreate new <URL_INSTALLER>`



I “Profili” di Delphi

- Nella documentazione è chiamato **Alternate registry key**
https://docwiki.embarcadero.com/RADStudio/Sydney/en/IDE_Command_Line_Switches_and_Options
- Fondamentale per lavorare con **versioni diverse** della stessa libreria
- In Delphi i package design-time sono veri e propri **plugin** dell'IDE



Workspace

- In blocks il workspace è una cartella legata ad una **versione e un profilo** di Delphi
- Comando `INIT`. Crea la cartella `.blocks` con:
 - configurazione, database dei pacchetti installati, copia del repository, `bpl/dcp`
- Comando `CONFIG`. Permette di visualizzare e modificare tutti i parametri di configurazione relativi al Workspace



Package e dipendenze

- In Delphi e package compilati producono due output:
 - **BPL**: Usati dall'IDE per i pacchetti design-time e dagli applicativi per applicazioni che usano i runtime packages. Vengono cercati nel path di sistema
 - **DCP**: usati in fase di compilazione di altri package. Il compilatore li cerca nel "library path" come i DCU

demo time





Package e dipendenze

- Impostazioni effettuate in **fase di compilazione**:
 - /p:DCC_BplOutput: i BPL vengono scritti in una directory dentro il Workspace
 - /p:DCC_DcpOutput: stessa cosa per i DCP
 - /p:DCC_DcuOutput: i dcu vengono messi in una directory dentro il pacchetto installato
 - /p:DCC_UnitSearchPath: aggiungo al search path la directory dei DCP



Package e dipendenze

- Durante l'**inizializzazione** (comando INIT):
 - Blocks modifica il Path locale usato da Delphi in modo da fargli trovare i BPL (*Tools / Options... / Environment*)
 - (Per usare i package runtime nell'applicazione è necessario modificare il path sulle impostazioni della macchina)



Creare Blocks

3



Il manifest

- Ogni “Block” deve avere il suo manifest
 - **Informazioni generali**: id, nome, versione, ...
 - **Compilazione e installazione**: package, versioni di Delphi, ...
 - **Dipendenze**: elenco dei Blocks dai quali dipende
- Per ogni versione deve essere prodotto un manifest



Il manifest

- Dati principali: id, name, version
- Repository: al momento supporta github e locale
- Platform: al momento testato win32 e win64
- Packages: elenco dei package da compilare
- Package folder: associazione tra folder e versione di delphi
- Dipendenze: elenco dei pacchetti con le versione. Viene rispettato l'ordine (*TOrderedDictionary*) e per le versioni viene usato **SemVer**



Semantic Versioning

- Dato un numero di versione MAJOR.MINOR.PATCH si incrementa:
 - MAJOR: in caso di **API non compatibili**
 - MINOR: per nuove funzionalità **retrocompatibili**
 - PATCH: per correzioni di **bug retrocompatibili**
- Etichette aggiuntive per pre-release e metadati di build sono disponibili come estensioni al formato MAJOR.MINOR.PATCH (per ora non supportato da Blocks)
- Nelle versioni 0.x.x, le MINOR possono essere incompatibili



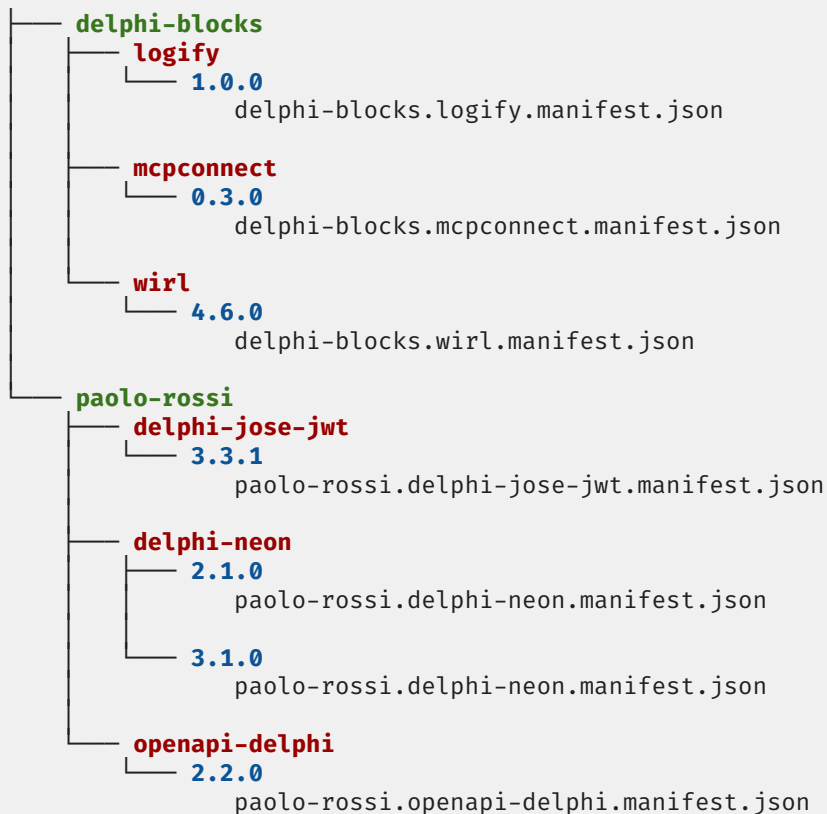
Semantic Versioning

Constraint	Meaning	Matches (example)
1.2.0	Exact version.	only 1.2.0
^1.2.0	Same major: $\geq 1.2.0$ $< 2.0.0$.	1.2.0, 1.9.9 — not 2.0.0
~1.2.0	Same minor: $\geq 1.2.0$ $< 1.3.0$.	1.2.0, 1.2.7 — not 1.3.0
$\geq 1.0.0$	At least 1.0.0.	1.0.0, 2.5.1
$< 2.0.0$	Below 2.0.0.	1.9.9 — not 2.0.0
$> 1.0.0$ / $\leq 1.0.0$	Strictly greater / less-or-equal.	—
$\geq 1.0.0$ $< 2.0.0$	Explicit range (space = logical AND).	1.4.0 — not 2.0.0
1.* / 1.2.*	Wildcard: any patch (or any minor) of the prefix.	1.* \rightarrow 1.7.3
* or empty	Any version.	anything

Manifest

```
{
  "id": "delphi-blocks.mcpconnect",
  "name": "MCPCConnect",
  "version": "0.3.0",
  "description": "Delphi Library for MCP",
  "repository": { "type": "github", "url": "https://github.com/..." },
  "platforms": {
    "Win32": { ... }
  },
  "packages": [{
    "name": "MCPCConnect",
    "type": ["runtime"]
  }],
  "packageOptions": { ... },
  "dependencies": {
    "paolo-rossi.delphi-neon": "^3.1.0",
    "delphi-blocks.logify": "^1.0.0"
  }
}
```

Manifest





Script

- Alcune volte può essere necessario eseguire del codice arbitrario
 - Gestione degli eventi: **install**, **uninstall**, **compile**
 - **before**, **after**
 - Al momento solo “copyres” e “echo”
- **Variabili d'ambiente:**
 - PACKAGE, PLATFORM, CONFIG, WORKSPACE_PATH, BPL_PATH

Script


```
"scripts": [  
  {  
    "description": "Copy resources and dfm",  
    "event": "afterCompile",  
    "command": "copyres"  
  },  
  {  
    "description": "Tell the user where the DCUs went",  
    "event": "afterCompile",  
    "command": "echo",  
    "args": ["Compiled %PACKAGE% for %PLATFORM%/%CONFIG%"]  
  }  
]
```

demo time



An aerial photograph of a city, likely Florence, Italy, showing a dense urban landscape with a river (Arno) winding through it. A large, historic square (Piazza della Signoria) is visible in the lower center, featuring a prominent building with arches. The image is overlaid with a semi-transparent dark layer to accommodate text.

DELPHIDAY


italian conference

THANK YOU