

# B4P

## Beyond Former Performance.

*A powerful programming language and analytics engine  
enabling rapid transformation of big data into powerful insights*

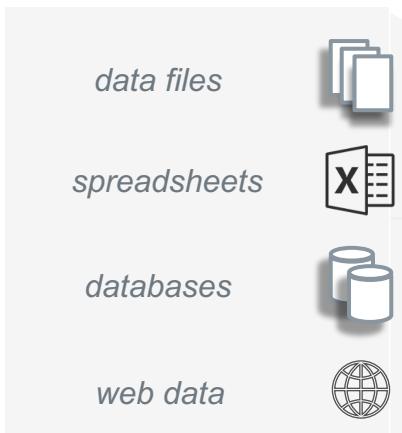
*Transforming Big Data into Powerful Insights*



# Overview

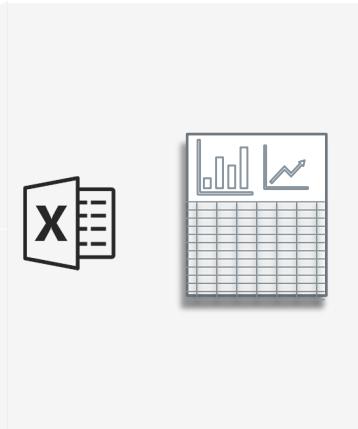
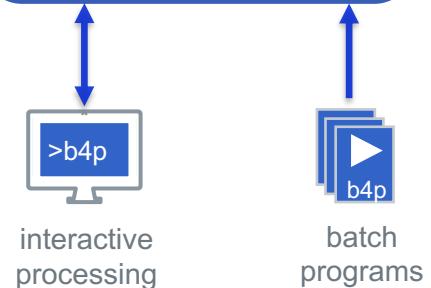
## The B4P Data Integration and Analytics Engine

Multiple complex  
continuously changing  
data sources



B4P Data Integration  
and Analytics Engine

Integrated analysis  
in seconds



data sources

- data files** Excel, XLS, CSV, XML, JSON, HTML, Zip, Text (and others)
- databases** Database exports (Salesforce, Oracle, SAP, Filemaker, et al)
- web data** Internet sources of structured data (websites, web services)
- other data** Statistical (R, SAS, SPSS, Stata), PDF (via Tabula)



## Table of Contents

---

1

### **Problem Statement**

2

### **Solution: Analytics and Execution Engine**

3

### **Real-world Use Cases**

4

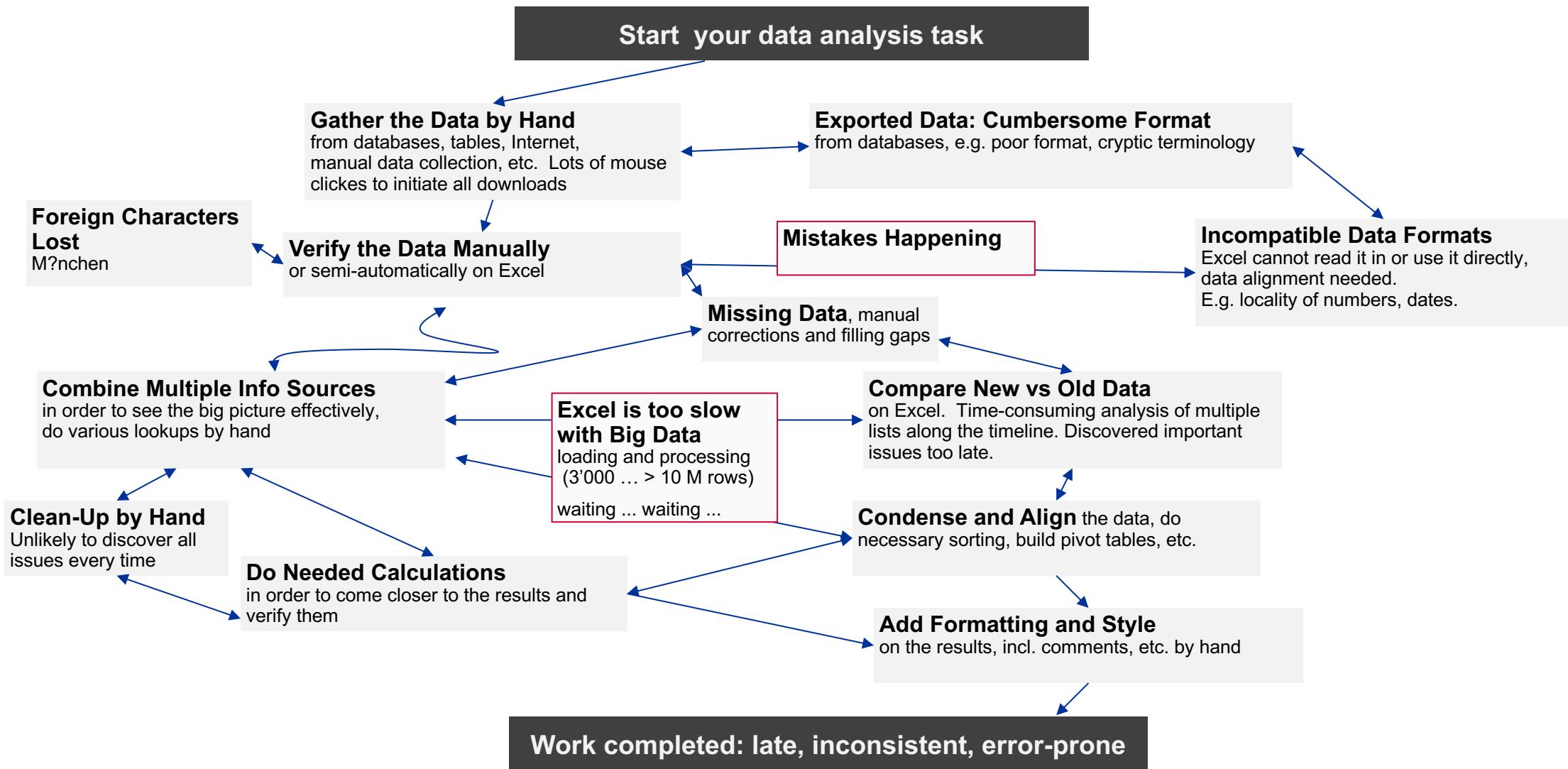
### **The Language**

5

### **Low-Code Solutions**

# Problem Statement

## Manual data integration and analysis is labor-intensive and error-prone



# Problem Statement

## Conventional methods of analysis automation are complex and unsustainable

---

### Write Excel Macros

(Visual Basic)

- OK for simple tasks, but ...
- ... coding becomes cumbersome if problems are more complex. Vulnerable if data format changes.
- Processing performance drops significantly when working with large data volumes.

**Poor performance and complex code if tasks are not small and simple**

### Write a Computer Program

(C, Java, Python, etc.)

- Runs fast, but takes a lot of time to program, debug and optimize.
- Others may have difficulties to understand what you have written.
- Such programs end up very large, with many functional details coded by hand.
- Good programming know-how, ideally object-oriented programming skills are needed, as well as obtaining a suitable development environment.

**Cryptic, unreadable code. Cannot be shared or adapted by business users.**

### Hire a Consultant

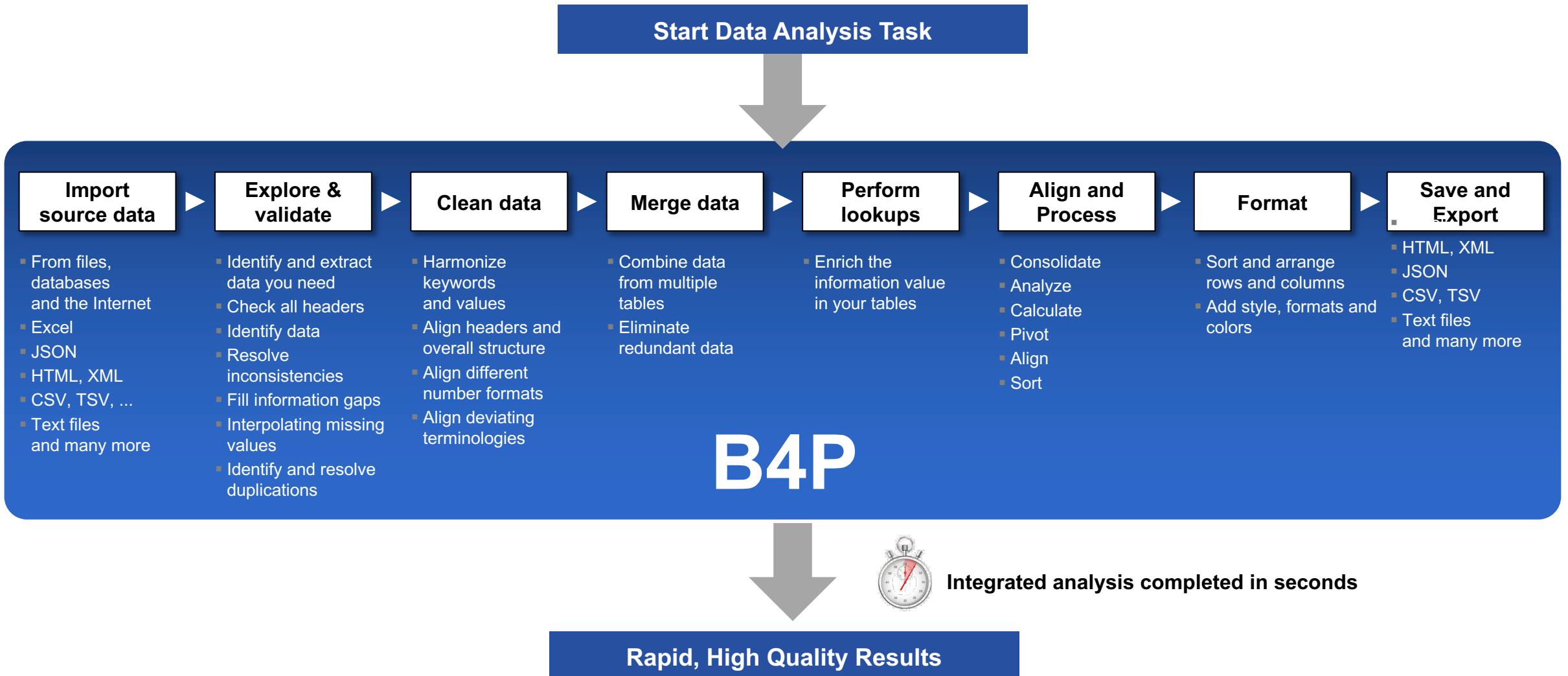
(or two)

- They are happy to solve your problems for cash. Solutions are quite decent, but ...
- ... if you need further enhancements, they will ask for more cash.
- You will depend on them as they expected, and keep convincing your boss to have these expenses approved.

**Expensive, external vendor dependency, no long-term sustainability**

# Solution

## Automate your data integration and analysis with a low-code analytics engine





## Table of Contents

---

1

**Problem Statement**

2

**B4P Analytics and Execution Engine**

3

**Real-world Use Cases**

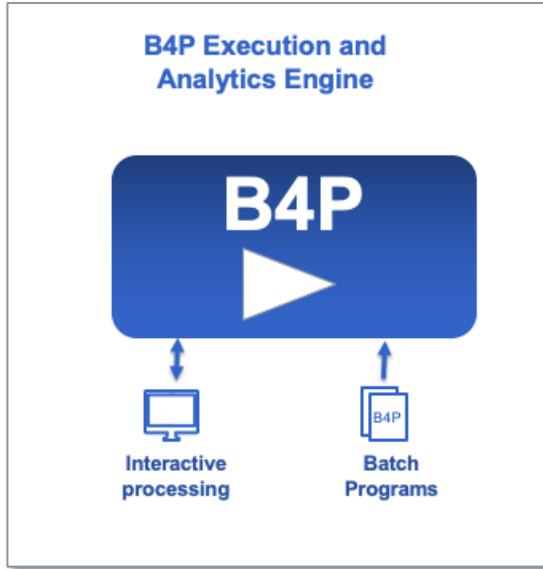
4

**The B4P Language**

5

**Programming Examples**

## The Engine



## The Language

```

table load excel file      ( football club, Football Membership List.xlsx );
table load                ( soccer club, Soccer Membership List.csv ); // Beginners = Novices
table rename column headers ( football club, { Family Name, City }, { Last Name, Town } );
table process selected rows ( soccer club, [Level]==Novice, [Level]=Beginner );
table merge                ( football club, soccer club, {Last Name,First Name},
                             {Level,Town}, append, " or " );
table sort rows            ( soccer club, { Level, Last Name, First Name });
table rearrange columns    ( soccer club, { Level, First Name, Last Name, Town } );
table save excel file      ( soccer club, Soccer Club, New Soccer Club Membership List.xlsx );
echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");

```

- Lean architecture of the engine delivers **full machine performance**
- Supports **various data formats** for inputs and outputs (Excel, HTML, XML, JSON, text files, etc., full UNICODE)
- Processes and delivers **accurate results reliably**
- High performance even with big data – **Matter of seconds, not hours**
- Styled and formatted output** for Excel and HTML  
(e.g. Structured tables, colors, multiple Excel sheets per file)

### Principle of Low-Code Approach: Few statements suffice

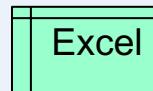
- A simple syntax: Easy to read, learn, understand and run
- Key strengths on large **structured data tables** and **hierarchically structured variables**
- Extensive library with very powerful functions and features
- Compact formulation methods for powerful processing steps minimizes coding loops and using variables

# Supported Data Formats

## Inputs

### Excel

- **XLSX**, **XSLM**, open formats
- **CSV** comma and tab separated files



### Database Exports

- **HTML**, **MHTML** and **XML** formats (depending what the database is producing). Examples: Salesforce, Oracle, SAP
- **JSON** files (JavaScript Object Notation format)
- **CSV** comma / tab / semicolon / ...symbol separated files

### Other Inputs

- Files with fixed columns on every row
- Any other form of structured text files
- **ZIP** files (B4P does data decompression)

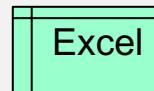
### Character Sets (both input and output)

- UNICODE UTF-8 and UTF-16; Basic and extended multilingual planes
- Legacy formats (like ASCII / Windows West Europe)

## Outputs

### Unformatted Output for Excel

- **CSV** comma separated files



### Formatted Output for Excel (with colors, formatting and style)

- **XLSX** (Excel 2007 onwards, in use today)
- **XLS** (Excel 2003 XML format)

### Unformatted and formatted output for Browsers

- **HTML** (incl. colors, formatting and style)
- **XML** (planned)

### Output for other databases

- **CSV** comma / tab / semicolon / ... symbol separated files
- **JSON** files
- Plain text files
- **ZIP** files (B4P does data compression)

Additional data formats can be supported easily using a B4P library extension



## Table of Contents

---

1

**Problem Statement**

2

**B4P Analytics and Execution Engine**

3

**Real-world Use Cases**

4

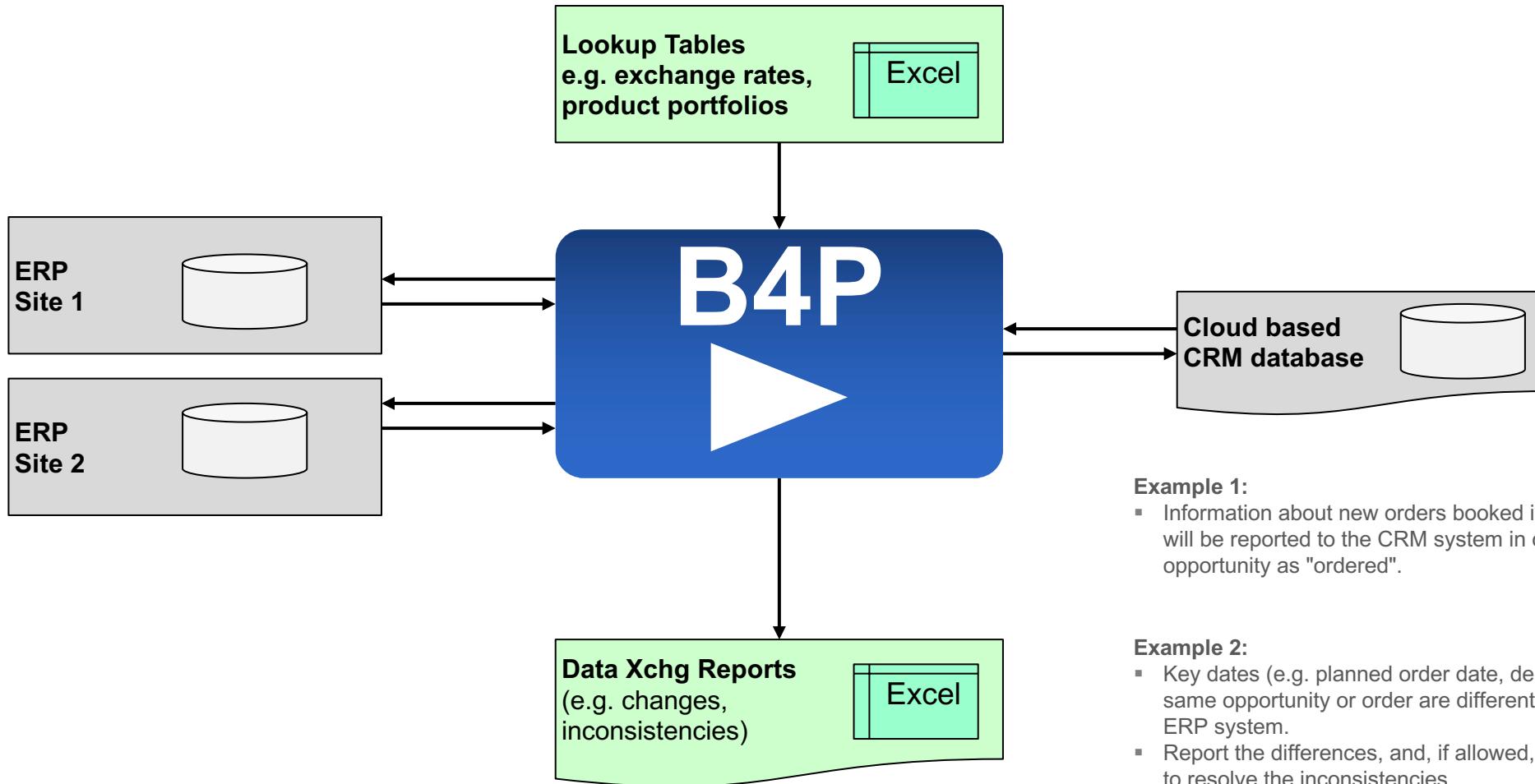
**The B4P Language**

5

**Programming Examples**

# B4P Real-world Use Case #1

## Information interchange between multiple different databases



### Example 1:

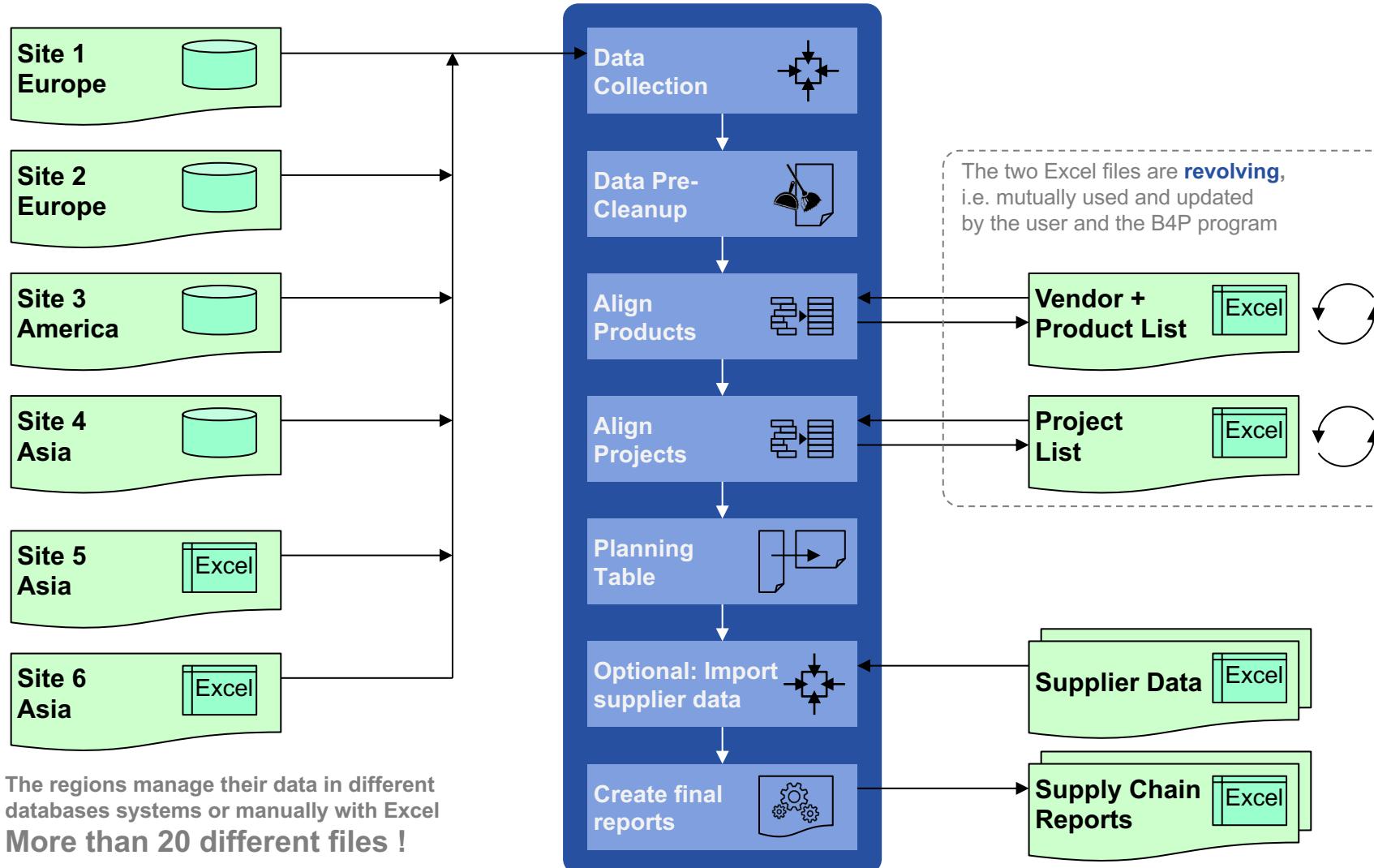
- Information about new orders booked into the ERP system will be reported to the CRM system in order to mark the opportunity as "ordered".

### Example 2:

- Key dates (e.g. planned order date, delivery dates) for the same opportunity or order are different in the CRM and ERP system.
- Report the differences, and, if allowed, use automated rules to resolve the inconsistencies

# B4P Real-world Use Case #2

## Heterogeneous data integration from branch offices worldwide



### 1. Load all Demand Files

- The data from the different sites originate from different database exports or manually prepared Excel files

### 2. First Clean-Ups

- Harmonize data formats to week numbers and years

### 3. Product Alignment

- The revolving table manages the products to include and allow for using harmonized product names.
- Orientation is by common product identification number.

### 4. Project Name Alignment

- Different project names and/or abbreviations are used by the sites. They will be aligned.

### 5. Project Name Alignment

- The sequential list of individual demands is transformed to a horizontal planning table with weekly schedule.
- Information consolidation and summing up

### 6. Import supplier planning data

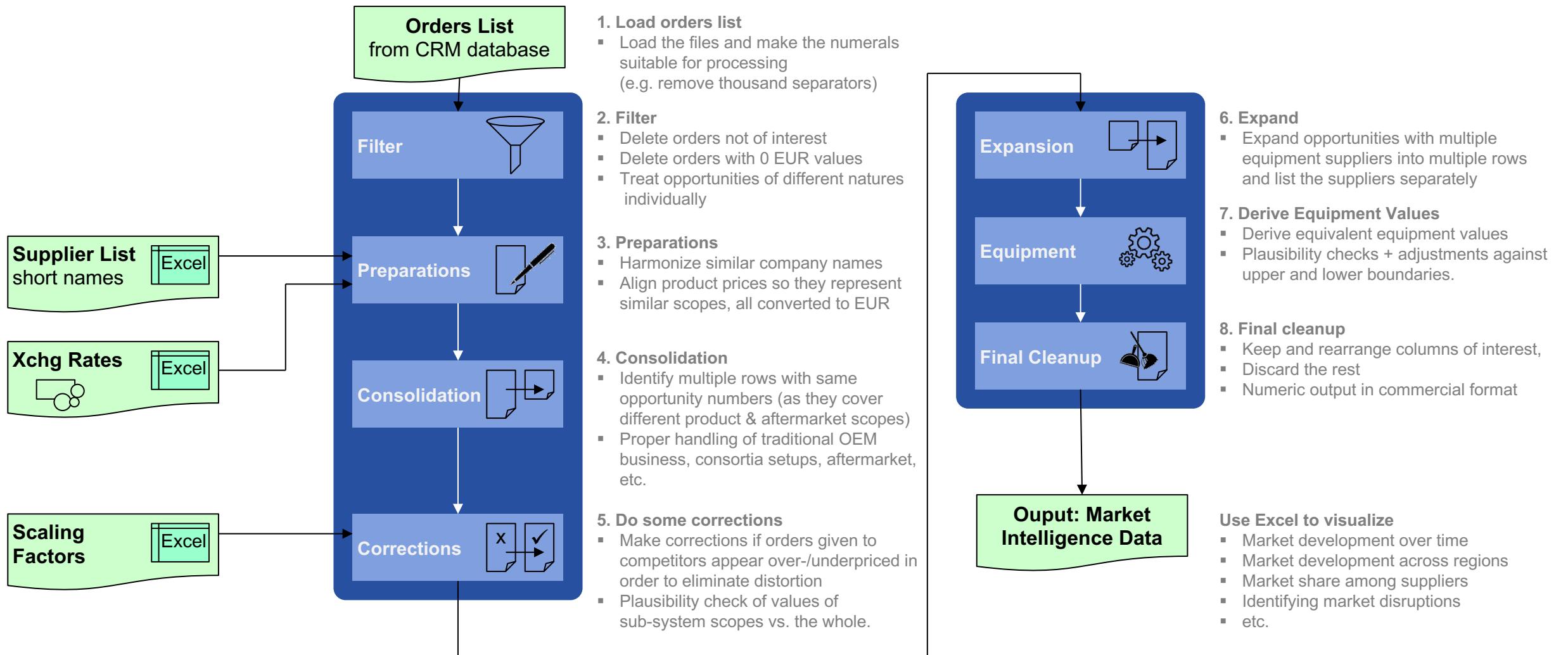
- If supply info is available, then match demand with their delivery plans

### 7. Reports

- Detailed internal reports for performance monitoring
- Condensed reports for suppliers

# B4P Real-world Use Case #3

## Enriched Business Intelligence





## Table of Contents

---

1

**Problem Statement**

2

**B4P Analytics and Execution Engine**

3

**Real-world Use Cases**

4

**The B4P Language**

5

**Programming Examples**

# B4P: The Language

## Key Benefits of a Low-Code Language Approach

---

- **Simple** procedural language
- **Easy to read and understand** the code, therefore very easy to learn programming.
- **Powerful language semantics keeps your program short to solve complex problem.**
- **Immediately** get your code running
- **Clear, natural language.** Give your variables, tables, functions, etc. natural names (spaces are allowed !)
- **Big Data Tables** of any size are one of the main data storage models and B4P is optimized for this.
- **No programming complexity** such as type definitions, declaring all the variables and doing memory management on your own.
- Significantly less need for fine grained programming like formulating loops, using variables, coding detailed algorithms, etc.
- B4P understands data formats such as **Excel, HTML, XML, JSON, CSV**, etc. to retrieve data from Excel, database and the Internet directly
- The execution engine and all library files are **very light-weight and lean**, very robust and start quickly.
- **Portability** (Windows, Linux, MacOS, etc.), enabling to run the same code on any computer.
- Output Excel files **with style and formatting** like colors, row widths, etc.
- **Rich function library** with **over 800 functions**, including 200 functions for processing tables, and growing.

The B4P Language allows you to express yourself easily in plain English to solve the problem.  
Focus on the *what*, not the *how*.

# B4P: The Language

## Language Syntax and Semantics

---

- Overall language block structure similar to C / C++ / Java.
- Full and homogeneous UNICODE support.
- Tables and structured variables are the two main data storage mechanisms.
- Table names, variable names and function names are fully flexible, e.g. multiple words and spaces are allowed.
  - Full Excel support, including formatting and style.
- Variables organized in a dynamic tree, allowing to build up nested arrays and structures.
  - Load / save sophisticated JSON contents to / from the variable structure using 1 statement.
- Code pieces can be passed as function parameters which will be executed multiple time or on a on-demand basis:
  - Example:
  - Benefit: Eliminates need to write loops or other details.
- Numerous flexible control flow mechanisms, going beyond the common ones like if, while, for, ...
- Cross platform compatibility: Windows / Linux / MacOS:
  - File names with directory paths are understood and interpreted correctly in other platforms (e.g. Windows vs. Linux).
  - Your program does not need to be modified to run on a different system.
- Powerful parameter set and matrix operations to process big data.

**"Think Big" when writing code – and do great things with 5-20 statements**

2. **Use deep operations** (vector and matrix operations) to process large amount of data inside tables and parameter sets.
3. Think how you can formulate your code in a very compact manner without compromising comprehensibility.



## Table of Contents

---

1

**Problem Statement**

2

**B4P Analytics and Execution Engine**

3

**Real-world Use Cases**

4

**The B4P Language**

5

**Programming Examples**

# B4P: Program Example 1

## Merging 2 Tables: Problem Statement

Football Membership List.xlsx

First Name	Family Name	City	Level
Abel	Amberstone	Amsterdam	Beginner
Beata	Berghill	Barcelona	Experienced
Corinne	Carlson	Copenhagen	Beginner
Dietmar	Davis	Dublin	Beginner
Ellen	Evans	Essen	Beginner
Fred	Fisher	Frankfurt	Experienced
Gregory	Green	Gaza City	Experienced
Henry	Hansson	Hamburg	Experienced
Ida	Ingelberg	Ingolstadt	Beginner
John	Janssen	Johannesburg	Beginner
Karl	Karlsson	Kansas City	Experienced

Soccer Membership List.xlsx

Level	Town	Last Name	First Name
Questionable	Kyoto	Karlsson	Karl
Novice	London	Lee	Linda
Experienced	Morristown	Miller	Mike
Experienced	New York	Nguyen	Nathali
Experienced	Oslo	Oliveiro	Oscar
Novice	Phoenix	Paulsson	Petra
Novice	Quebec City	Quarles	Quincy
Experienced	Riga	Richardson	Richard
Experienced	San Diego	Stewart	Sandra
Experienced	Tahoma	Turner	Tim
Questionable	Ulm	Ufford	Uwe
Novice	Venice	Viking	Victor



New Soccer Membership List.xlsx

Level	First Name	Last Name	Town
Beginner	Abel	Amberstone	Amsterdam
Beginner	Corinne	Carlson	Copenhagen
Beginner	Dietmar	Davis	Dublin
Beginner	Ellen	Evans	Essen
Beginner	Ida	Ingelberg	Ingolstadt
Beginner	John	Janssen	Johannesburg
Beginner	Linda	Lee	London
Beginner	Petra	Paulsson	Phoenix
Beginner	Quincy	Quarles	Quebec City
Beginner	Victor	Viking	Venice
Experienced	Beata	Berghill	Barcelona
Experienced	Fred	Fisher	Frankfurt
Experienced	Gregory	Green	Gaza City
Experienced	Henry	Hansson	Hamburg
Experienced	Mike	Miller	Morristown
Experienced	Nathali	Nguyen	New York
Experienced	Oscar	Oliveiro	Oslo
Experienced	Richard	Richardson	Riga
Experienced	Sandra	Stewart	San Diego
Experienced	Tim	Turner	Tahoma
Questionable	Uwe	Ufford	Ulm
Questionable or Experienced	Karl	Karlsson	Kyoto or Kansas City

A new football club should be created after merging two existing sports clubs:

- The tables of the two clubs are arranged differently and contain different acronyms (e.g. qualification levels)
- Some people are members in both clubs and need to be resolved properly.
- Highlight possible inconsistencies (red text color)

# B4P Example: Low-code Solution

## 8 statements: load, clean, align semantics, merge, and save

Simple statements  
easy to understand

Excel file loaded with  
a single simple statement

No loops. Simple statements  
apply for whole tables

One statement merges  
two tables as specified

Full Excel support

Multiple words for functions, variables, table names, header names, allow for readability and naming flexibility

```
table load excel file  
table load  
table rename column headers
```

```
table process selected rows
```

```
table merge
```

```
table sort rows  
table rearrange columns
```

```
table save excel file
```

```
echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");
```

```
( football club, Football Membership List.xlsx );  
( soccer club, Soccer Membership List.csv ); // Beginners = Novices  
( football club, { Family Name, City }, { Last Name, Town } );
```

```
( soccer club, [Level]==Novice, [Level]=Beginner );
```

```
( football club,  
soccer club, {Last Name,First Name},{Level,Town},append," or " );  
( soccer club, { Level, Last Name, First Name } );  
( soccer club, { Level, First Name, Last Name, Town } );
```

# B4P Example: Low-code Solution

## 7 additional statements to add style to the Excel file

Simple statements  
easy to understand

Simple and target inde-  
pendent approach to format  
tables, rows, columns and cells

Full Excel support

Highly comprehensible function names

```
table style table
table style rows
table style columns

table process selected rows
    table style cells

translate style attributes for excel( soccer club );
table save excel file
echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");
```

```
( soccer club, sheet, column width, 20, row height, 20,
    freeze rows, 1, autofilter, 0, vertical align, center );
( soccer club, 0, sheet, boldface, true, fill color, gray 15 );
( soccer club, Level, sheet, column width, 30 );

( soccer club, ([Level] = '*Questionable*'),
    ( soccer club, Level, row(), single, text color, red ) );
```

# B4P: Program Example 2

## U.S. Presidents from Wikipedia

destroy the fragile unity holding the nation together, Washington remained unaffiliated with any political faction or party throughout his eight-year presidency.<sup>[2]</sup>

Contents [hide]  
1 Presidents  
1.1 President-elect  
2 Subsequent public office  
3 See also  
4 Notes  
5 References  
6 External links

### Presidents

	Presidency <sup>[a]</sup>	President	Party <sup>[b]</sup>	Election	Vice President
1	April 30, 1789 – March 4, 1797	George Washington	Unaffiliated	1788–89 1792	John Adams <sup>[c]</sup>
2	March 4, 1797 – March 4, 1801	John Adams	Federalist	1796	Thomas Jefferson <sup>[d]</sup>
		Aaron Burr		1800	



President	Presidency (1)	President	Party	Election	Vice President
	1 April 30, 1789 – March 4, 1797	George Washington	Unaffiliated	1788–89, 1792	John Adams, Thomas Jefferson
	2 March 4, 1797 – March 4, 1801	John Adams	Federalist	1796	
	3 March 4, 1801 – March 4, 1809	Thomas Jefferson	Democratic-Republican	1800, 1804	Aaron Burr, George Clinton
	4 March 4, 1809 – March 4, 1817	James Madison	Democratic-Republican	, Vacant after Apr. 20, 1812, 1808, 1812	Elbridge Gerry, Vacant after Nov. 23, 1814
	5 March 4, 1817 – March 4, 1825	James Monroe	Democratic-Republican	1816, 1820	Daniel D. Tompkins,
	6 March 4, 1825 – March 4, 1829	John Quincy Adams	Democratic-Republican	1824,	John C. Calhoun, , Vacant after Dec. 28, 1832,
	7 March 4, 1829 – March 4, 1837	Andrew Jackson	Democratic	1828, 1832	Martin Van Buren
	8 March 4, 1837 – March 4, 1841	Martin Van Buren	Democratic	1836	Richard Mentor Johnson
	9 March 4, 1841 – April 4, 1841	William Henry Harrison	Whig	1840	John Tyler
	10 April 4, 1841 – March 4, 1845	John Tyler	Whig	Vacant throughout presidency,	
	11 March 4, 1845 – March 4, 1849	James K. Polk	Democratic	1844	George M. Dallas

Download the list of Presidents and generate Excel table with one president per row

- Data source: [https://en.wikipedia.org/wiki/List\\_of\\_presidents\\_of\\_the\\_United\\_States](https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States)
- Some Presidents had multiple terms
- Ignore the portraits
- Some vice presidents had deviating terms
- Remove redundant artefacts, e.g. cross-referencing symbols
- Generate a nice table with **parties colored differently**

# B4P: Program Example 2

## 16 Statements, 0 Loops and 0 Variables to Straighten up the Presidents

Internet Download  
in 1 statement

Cleanups  
with minimum effort

Align the data.  
Contents done!

Define colors  
affiliated to the parties

Color and Style the results

Save as Excel File

```
include ( Style Library ); // This library needs to be included if you want to add style and formatting to the table

file download overwrite      ( "https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States", presidents.html );
table load                   ( presidents, presidents.html, HTML, 'id="Presidents"' );

// Clean up and strip out all footnote references and new lines in the fields
table delete rows           ( presidents, table length( presidents ) -1 ); // Remove last row with redundant footnote info
table process all cells      ( presidents, [.] = replace all( literal([.]), { "[a]" .. "[z]", new line}, '' ) );

// Remove the blank column originally containing portraits and put president name into all rows

table delete columns         ( presidents, {President, Party} );
table rename column headers  ( presidents, {"President (1)", "Party (1)"}, {President, Party} );
table fill vertically        ( presidents, President );
table consolidate            ( presidents, President, { Election, Vice President }, append, ", " );

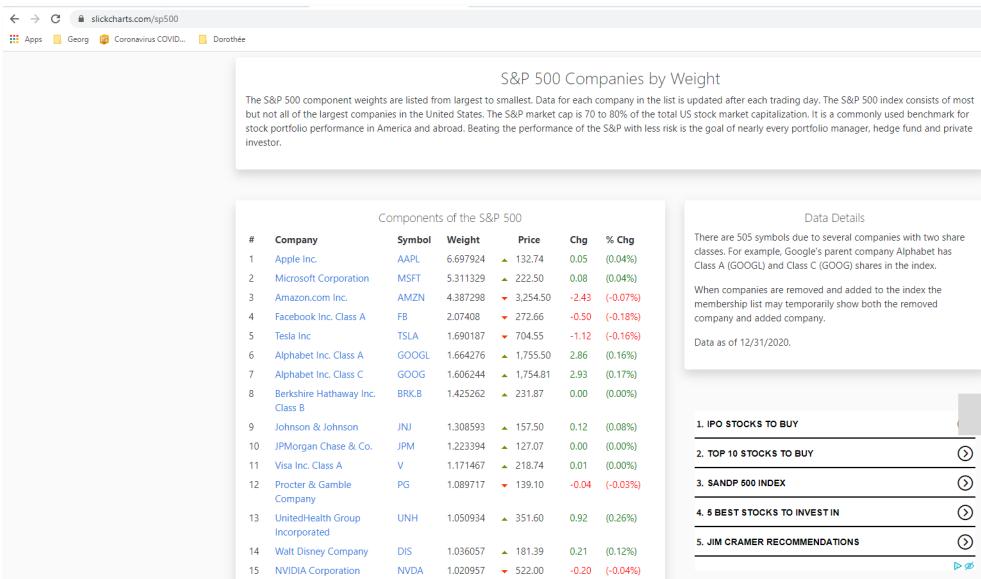
table initialize ( party colors,    // Define party colors
{{ Party Name,                  Colors },
{ Democratic,                  "#8080FF" }, // Light blue
{ Republican,                 "#FF8080" }, // Light red
{ Federalist,                  coral },
{ "Democratic- Republican",   excel light green },
{ "Democratic-Republican",    excel light green },
{ Whig,                         yellow },
{ National Union,              ocre },
{ Unaffiliated,                gray 15 } } );

// Add some colors and styles
table process                 ( presidents, table style cells( presidents, Party, row(), single,
fill color, [ party colors : Party Name, [Party], Colors ] ) );
table style columns            ( presidents, { "Presidency (1)", "President", "Vice President" }, sheet, column width, 30 );
table style columns            ( presidents, { Party, Election }, sheet, column width, 20, horizontal align, middle );
table style rows               ( presidents, 0, table, boldface, true );
table style table              ( presidents, sheet, wrap text, true, autofilter, 0, freeze rows, 1 );

translate style attributes for excel ( presidents );
table save excel file          ( presidents, All U.S. Presidents, presidents.xlsx );
```

# B4P: Program Example 3

## Stock Data (SP 500 and NASDAQ 100) Combined



#	Company	Symbol	Price	Chg	% Chg	Weight sp500	Weight nasdaq100
1	3M Company	MMM	175	0.21	0.12%	0.318459	
2	A. O. Smith Corporation	AOS	54.82	0	0.00%	0.023472	
3	ABIOMED Inc.	ABMD	326.6	2.4	0.74%	0.046276	
4	AES Corporation	AES	23.7	0.2	0.85%	0.049417	
5	AMETEK Inc.	AME	120.94	0	0.00%	0.087884	
6	ANSYS Inc.	ANSS	363.8	0	0.00%	0.098729	0.258
7	ASML Holding NV	ASML	486.99	-0.73	-0.15%		0.305
8	AT&T Inc.	T	28.76	0	0.00%	0.647313	
9	AbbVie Inc.	ABBV	107.15	0	0.00%	0.597494	
10	Abbott Laboratories	ABT	109.49	0	0.00%	0.612924	
11	Accenture Plc Class A	ACN	261.21	0	0.00%	0.522713	
12	Activision Blizzard Inc.	ATVI	92.85	0	0.00%	0.226706	0.593
13	Adobe Inc.	ADBE	500.1	-0.02	0.00%	0.757774	1.984
14	Advance Auto Parts Inc.	AAP	157.51	0	0.00%	0.033788	
15	Advanced Micro Devices Inc.	AMD	91.66	-0.05	-0.05%	0.348385	0.912
16	Aflac Incorporated	AFL	44.47	0	0.00%	0.091757	
17	Agilent Technologies Inc.	A	118.49	0	0.00%	0.114563	
18	Air Products and Chemicals Inc.	APD	273.22	0	0.00%	0.190791	

526	Zoom Video Communications Inc	ZM	339.2	1.88	0.56%		0.543
527	eBay Inc.	EBAY	50.11	-0.14	-0.28%	0.103937	0.286
528	salesforce.com inc.	CRM	222.53	0	0.00%	0.643119	

### Import the SP 500 and NASDAQ 100 listings and merge them

- Data source1: <https://www.slickcharts.com/nasdaq100>
- Data source2: <https://www.slickcharts.com/sp500>
- Some companies are listed in only one of them, others are listed in both.
- Combine the information, show the weighting in the two listings and color the stock price developments

# B4P: Program Example 3

## 13 Statements, 1 loop and 1 variable do the Job

Download 2 files  
and align header names

```
// Download two web pages.  
for all parameters ( {nasdaq100, sp500} , listing[] )  
{  
    file download overwrite      ( "https://www.slickcharts.com/" + listing[], listing[] + .html );  
    table load  
    table rename column headers  ( listing[], listing[] + .html, HTML, "Components of the" );  
    table save  
}  
// Weight info is specific to SP and Nasdaq, so add the listing name  
( listing[], listing[] + " as loaded.csv" );
```

Merge the 2 tables  
and assign common name

```
// Combine the two tables  
table merge extend columns  ( nasdaq100, sp500, Symbol );  
table rename  
( sp500, stocks );
```

Clean Up

```
// Cleanup  
table correct headers        ( stocks, '*Price*', Price );  
table rearrange columns       ( stocks, { '#', Company, Symbol, Price, Chg, '% Chg' } ); // Weightings follow afterwards  
  
// Percent value gets converted to regular number, price value is cleaned up  
table process                ( stocks, [% Chg] = smart numeral( middle( [% Chg], '(', ')' )); [Price] = clean numeral([Price]) );  
table sort rows               ( stocks, Company );  
table process                ( stocks, ['#'] = row() );
```

Save as Excel File

```
// Save the work  
table save excel file         ( stocks, "NASDAQ and SP500", Stocks.xlsx );
```

# B4P: Program Example 3

## 8 additional statements for style and color

Download Files

and align header names

```
include ( Style Library );

// Download two web pages.
for all parameters ( {nasdaq100, sp500} , listing[] )
{
    file download overwrite      ( "https://www.slickcharts.com/" + listing[], listing[] + .html );
    table load                   ( listing[], listing[] + .html, HTML, "Components of the" );
    table rename column headers ( listing[], "Weight", "Weight " + listing[] );
    // Weight info is specific to SP and Nasdaq, so add the listing name
    table save                   ( listing[], listing[] + " as loaded.csv" );
}

// Combine the two tables
table merge extend columns   ( nasdaq100, sp500, Symbol );
table rename                  ( sp500, stocks );

// Cleanup
table correct headers        ( stocks, '*Price*', Price );
table rearrange columns       ( stocks, { '#', Company, Symbol, Price, Chg, '% Chg' } ); // Weightings follow afterwards

// Percent value gets converted to regular number, price value is cleaned up
table process                ( stocks, [% Chg] = smart numeral( middle( [% Chg], '(', ')' )); [Price] = clean numeral([Price]) );
table sort rows               ( stocks, Company );
table process                ( stocks, ['#'] = row() );

// Add some color and formatting
table process                ( stocks,
    table style cells        ( stocks, {'Chg','% Chg'}, {2:row()}, single, text color, select if ( [Chg]>0, excel green, red ) ) );
    table style rows          ( stocks, 0, table, fill color, gray 14, boldface, true, wrap text, true );
    table style columns        ( stocks, Company, sheet, column width, 30 );
    table style columns        ( stocks, '% Chg', sheet, number format, "0.00%" ); // Value to show as percent.
    table style table          ( stocks, sheet, freeze rows, 1, autofilter, 0 );

// Save the artwork
translate style attributes for excel ( stocks );
table save excel file         ( stocks, "NASDAQ and SP500", Stocks.xlsx );
```

Clean Up

Add style and color

Save as Excel File

# B4P

## Beyond Former Performance.

Information

[www.b4p.app](http://www.b4p.app)

Contact

**Europe**

Georg zur Bonsen  
+41 56 221 82 00  
zur-bonsen (at) bluewin .ch

**North America**

Rafael M Richards  
+1 202 469 15 27  
rmrich5 (at) gmail .com

## Table of Contents

---

1

**The Problem Statement**

2

**The Analytics and Execution Engine**

3

**Typical Use Cases**

4

**The B4P Language**

5

**Programming Examples**

6

**Backup Slides**

# B4P: Beyond Former Performance

## New in Release 8.00

### 100% Excel Support

- B4P loads and saves the latest Microsoft Excel file format (.xlsx)
- Formatting and styles are supported
- B4P also supporting the old Excel XML 2003 file format
- Round trip: Generated excel files can be loaded again without losing data.

Excel files are actually ZIP files containing various files describing workbooks, tables, styles and shared strings.

The functions

- **table load excel file** and
- **table save excel file**

are implemented in B4P.

### Overall Clean-up

- Thorough functional tests and stress tests have been carried out to make B4P become even more reliable.
- The start-up behavior has been streamlined.

A small number of new functions have been added which include the following:

- Processing table columns
- Processing table cells
- Advance table search and update functions
- Improved visualization of tables
- Improved listing of tables, functions and variables

### Help and Documentation

- Moved from PDF to 100% online
- [www.b4p.app](http://www.b4p.app)
- Online help available inside B4P on your fingertips, i.e. by typing "docs"
- Lots of reproducible program examples
- Various new help features are available

The document generation has been automated using a B4P program. It fishes all raw contents from source code comments and supplementary text files, generates a master file (JSON) and finally produces all web document contents automatically.

Almost all programming examples are tested automatically with every new update and the outputs are included in the documentation.

# B4P Use Case

## Automatic documentation generation for website [www.b4p.app](http://www.b4p.app)

### Inputs: Source Text (inside C program)

```
//  sqrt
//
/* B4PDOCU.START

"sqrt - Square Root" :
{
    "Function Names": "sqrt",
    "Keywords": "Square Root",
    "Documentation": "Function Description",
    "Description": [
        "Calculates the square root of a value"
    ],
    "Call as": "function",
    "Parameter count": "1",
    "Restrictions": "Indirect parameter passing is disabled",
    "Parameters": [
        {
            "Number": "1",
            "Name": "Value",
            "Direction": "input",
            "Types": "numeral",
            "Description": "Value to use for logarithmic calculation"
        }
    ],
    "Return value": [
        {
            "Name": "Result",
            "Types": "numeral",
            "Description": "Calculated square root value"
        }
    ],
    "Exceptions": "Negative square root",
    "Examples": [
        "echo( sqrt( 4 ) );",
        "echo( sqrt( 2 ) );"
    ],
    "Output": "automatic",
    "See also": "Power Functions"
}
B4PDOCU.STOP */

void function_sqrt(int bias, Token_List& p, Token_Entry& ret_val)
{
    ...
    return;
}
```

### Output: Web contents



The screenshot shows a web page generated from the source code. The title is "sqrt - Square Root". The page includes a sidebar with a "Function Library Guide" containing links to various sections like User Guide, Language Guide, Table of Contents, and a detailed tree view of the Function Library. The main content area displays the function's name, description, call as, restrictions, parameters, return value, exceptions, examples, output, and see also sections. Each section contains detailed descriptions and tables for parameters and return values. The page has a consistent structure with a header and footer. Callouts on the right side highlight features: "Consistent structure", "Clean tables", "Program examples executed automatically", and "Smart cross-referencing".

**B4P**  
Beyond Former Performance.

**sqrt - Square Root**

Prev Next

**Function Library Guide**

- User Guide
- Language Guide
- Table of Contents
- 1 Function Library
- + 2 Control Flow Functions
- + 3 Console I/O Functions
- + 4 Type Conversion and Formatting
- + 5 Mathematics and Statistics
- + 5.1 Basic Math Functions
- + 5.2 Transcendental Functions
  - 5.2.1 sqrt - Square Root
  - 5.2.2 pow - Power Function
  - 5.2.3 Exponential Functions
  - 5.2.4 Logarithmic Functions
  - 5.2.5 Trigonometric Functions
  - 5.2.6 Hyperbolic Functions
- + 5.3 Series Functions
- + 5.4 Matrix Functions
- + 5.5 Statistics Functions
- 5.6 Conditional Combination Functions
- + 5.7 Finance and Business Functions
- + 6 String Functions
- + 7 Date and Time Functions
- + 8 Parameter Set Functions
- + 9 Table Functions
- + 10 Variables Functions
- + 11 Directory and File System Functions
- + 12 Network Functions
- + 13 Report Functions
- + 14 General Info

**Function Names**  
sqrt

**Description**  
Calculates the square root of a value

**Call as:** function

**Restrictions**  
Indirect parameter passing is disabled

**Parameter count**  
1

**Parameters**

No.	Type	Description
1	numeral	Value Value to use for logarithmic calculation

**Return value**

Type	Description
numeral	Result Calculated square root value

**Exceptions**  
Negative square root

**Examples**  
`echo( sqrt( 4 ) );  
echo( sqrt( 2 ) );`

**Output**  
2  
1.4142135624  
Try it yourself: Open LIB\_Function\_sqrt.b4p in B4P\_Examples.zip. Decompress be

**See also**  
Power Functions

**Consistent structure**

**Clean tables**

**Program examples executed automatically**

**Smart cross-referencing**

# B4P Use Case

## Automatic Document Generation for [www.b4p.app](http://www.b4p.app) using B4P

### Catalog of Books

- Short Excel File
- Specifies all manuals to generate
- Specifies location of source text for the different manuals

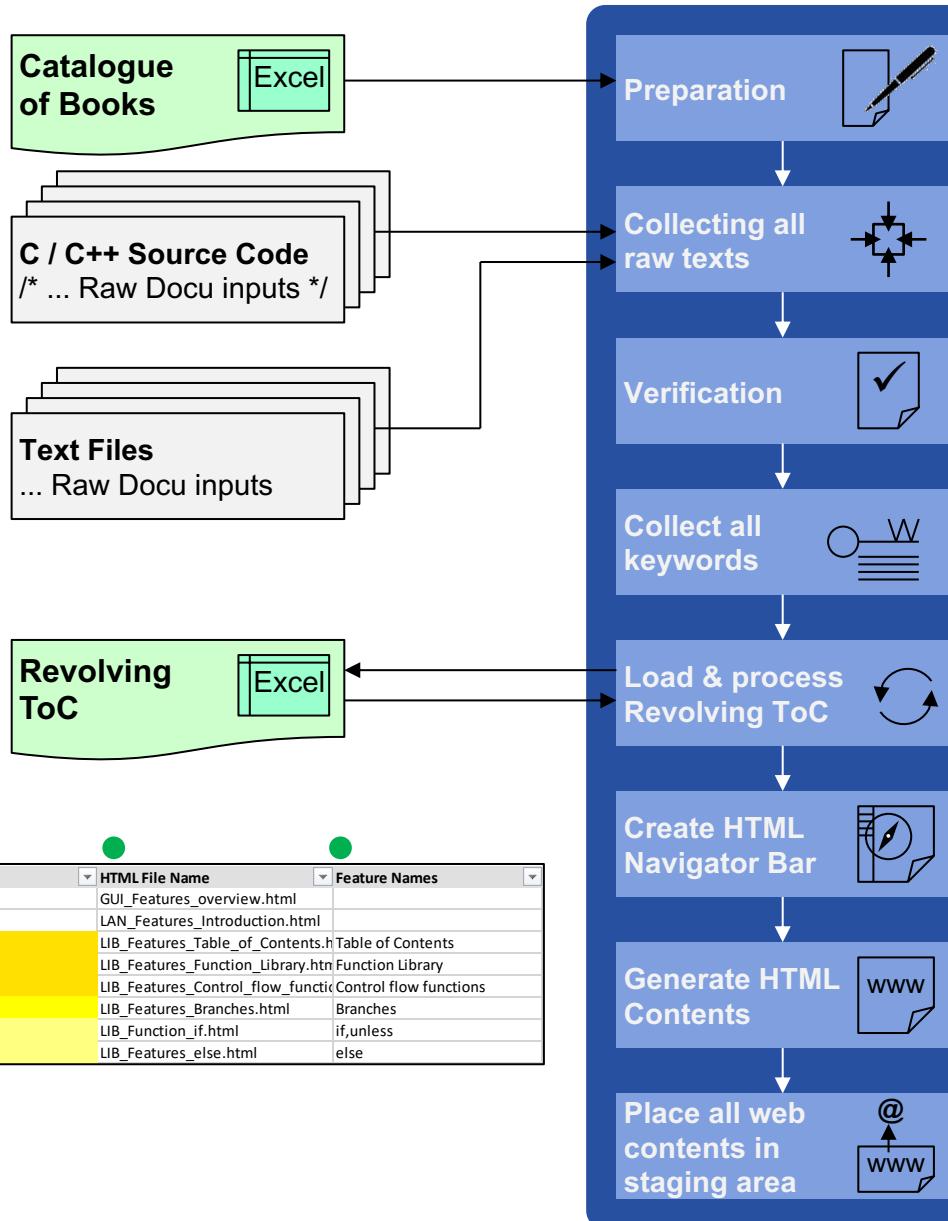
### Raw Inputs

- Descriptions of B4P functions are documented in the C/C++ files in comments, using an enhanced JSON format.
- Other contents such as introductory parts are described in additional text files, also using enhanced JSON format

### Revolving Table of Contents

- It puts all individual raw document sections into a given order and hierarchy level in the document
- Both user ● and B4P program ● update this table mutually

Status	Category	Level	Section Name	HTML File Name	Feature Names
Missing, (Ass Link)			&#8594; User Guide	GUI_Features_overview.html	
Missing, (Ass Link)			&#8594; Language Guide	LAN_Features_Introduction.html	
OK, (Assign TOC)	1	Table of Contents		LIB_Features_Table_of_Contents.h	Table of Contents
OK	Body	1 B 1	Function Library	LIB_Features_Function_Library.htm	Function Library
OK	Body	1 B 2	Control Flow Functions	LIB_Features_Control_flow_functions.htm	Control flow functions
OK	Body	2 B 2.1	Branches	LIB_Features_Branches.html	Branches
OK	Body	3 B 2.1.1	if, unless	LIB_Function_if.html	if,unless
OK	Body	3 B 2.1.2	else	LIB_Features_else.html	else



### Preparation

- Load catalogue of books
- User selects the book to generate

### Text Collection

- Scan all files in specified subdirectories for relevant contents for B4P documentation

### Master File JSON

### Verification

- Ensures that contents provided fulfill the structural guidelines

### Collect all Keywords

- Structured handling of all keywords and function names allowing convenient cross referencing and index pages

### Process Revolving ToC

- Add titles of new contents into placeholders or at the bottom
- Add further info (links, keywords)
- Do chapter/section renumbering

### Generate HTML Navigation Bar

- Left-hand menu to select section to see

### Generate HTML contents

- Formatted text and tables
- Pictures included
- **Execute all B4P program examples automatically and add their outputs into the doc contents**

### Staging Area

- All files (HTML, JPG, style.css, PDF, etc.) are moved to the staging area, ready for one-mouseclick publication on the Internet: [www.b4p.app](http://www.b4p.app)