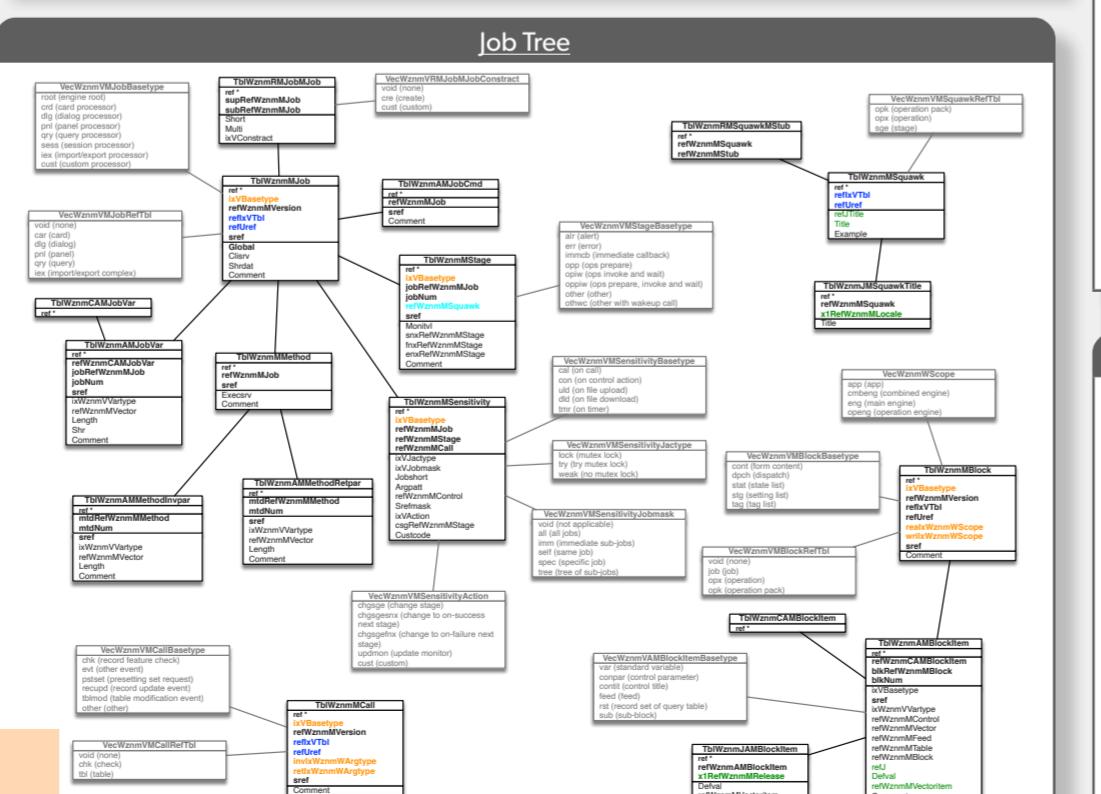
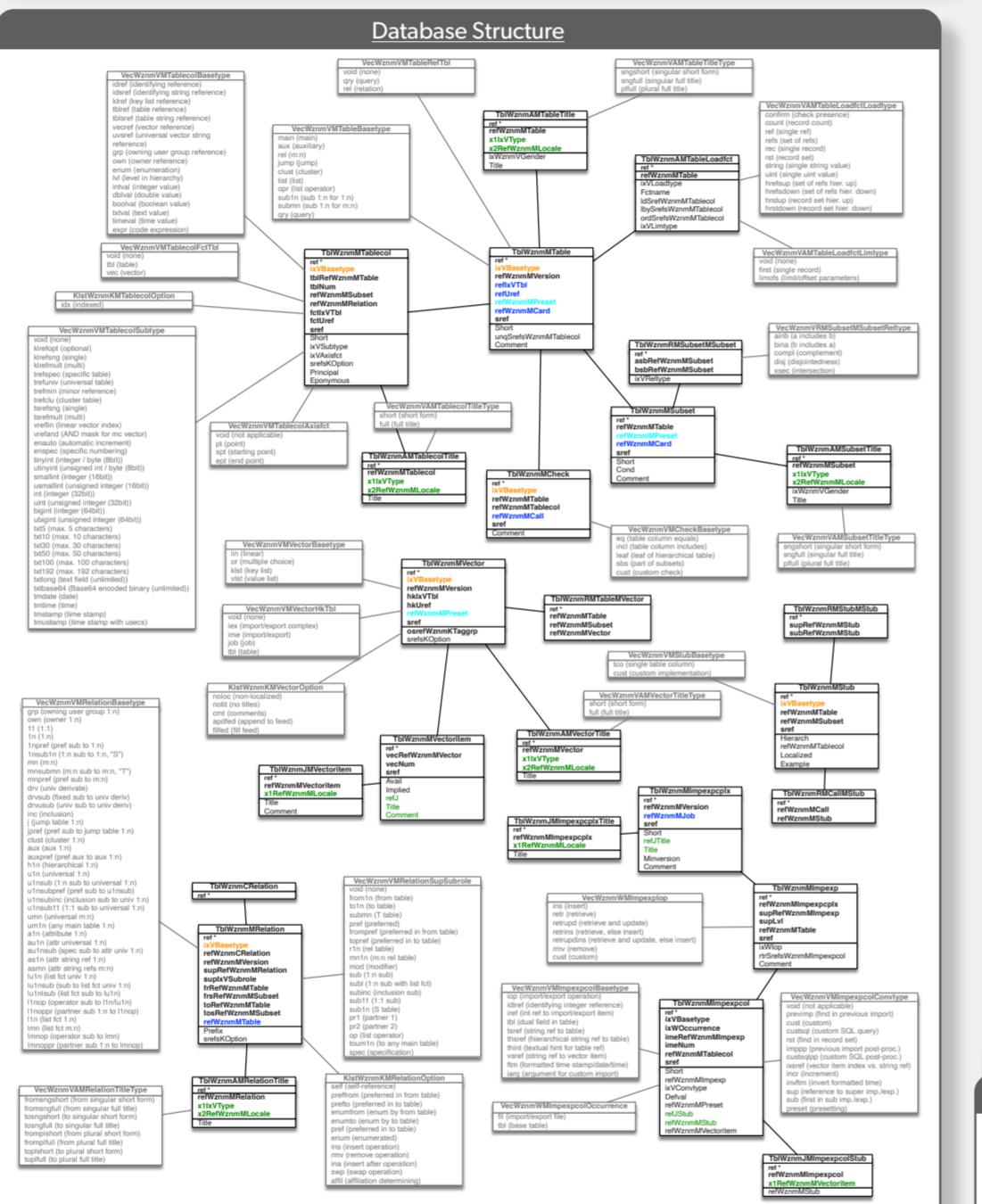
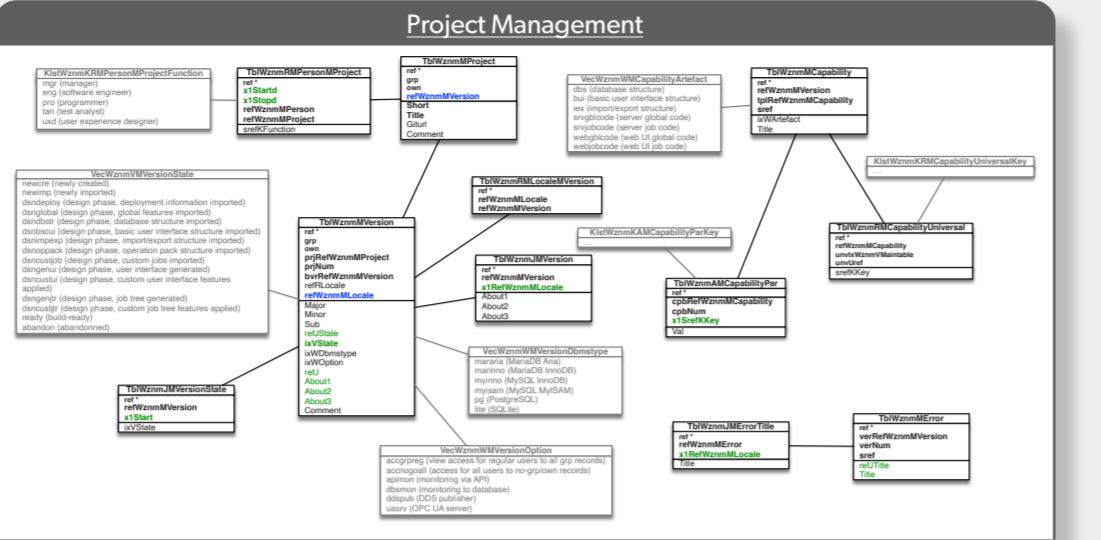
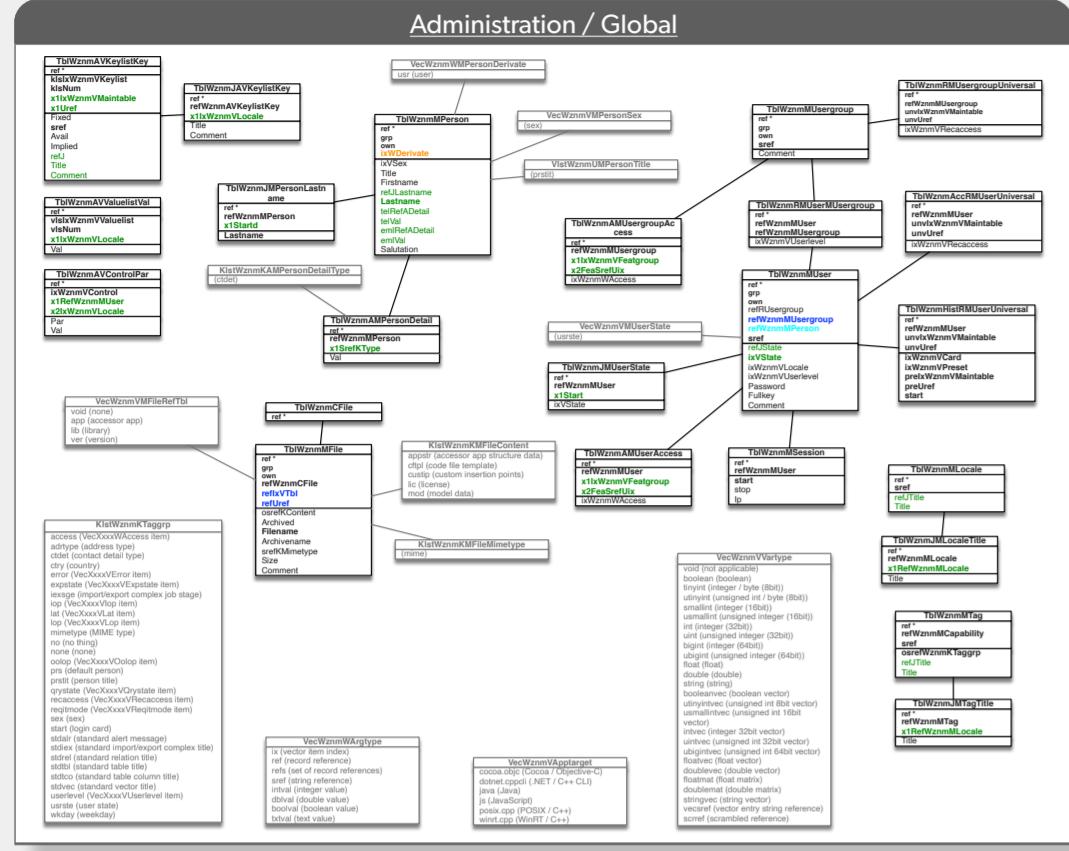


The WhizniumSBE Master Database

WhizniumSBE uses a comprehensive relational database structure into which model information is imported. Supported backends include MariaDB, MySQL and PostgreSQL. WhizniumSBE algorithms analyze the imported data and derive additional required database entries. Once all information is compiled in the database for a specific version of a WhizniumSBE-backed project, writing of source code can take place. This operation is based on the version's completed representation in the database on one hand, and on the previous version's source code tree – if available – on the other hand.

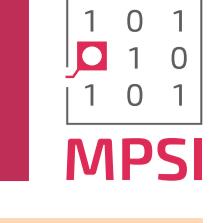


How to read the database diagram

- black boxes represent database tables, with their table columns listed sequentially
 - lines between database tables represent e.g. 1:N relations, typically top-down; only the most relevant relations are shown
 - gray boxes represent vectors / key lists (lists of options) with database / table scope

Minizinc SBE Cheat Sheet

Master The Service Builder's Edition Create-Rich Embedded Linux Applications



orial videos and more on the MPSI YouTube channel:
<https://content.mpsitech.cloud/youtube.html>



oles and tool repositories on MPSI's GitHub account:
<https://github.com/mpsitech>



MinizniumSBE Glossary Of Terms

um.mpsitech.cloud
hizniumDBE / SBE (scenario 2)

Workgroup development root
\$(HOME) / whiznium

workgroup server
runs: WhizniumDBE / SBE (scenario 1)

\$(WHIZDEVREROOT)
\$(HOME) / whiznium_dev (optional) NFS mount

\$(WHIZSDKROOT)
\$(HOME) / whiznium_sdk

\$(WHIZROOT)
\$(HOME) / whiznium

\$(WHIZSDKROOT) & \$(WHIZROOT) embedded target 1
\$(HOME) / boards/mytarget1/sdk/home/root/whiznium_sdk
\$(HOME) / boards/mytarget1/sdk/home/root/whiznium/_/web

oper workstation
DE e.g. VSCode, compiler / linker e.g. gcc, Java-based tools, (remote) debuggers
ins: locally deployable WhizniumSBE projects

Deployment root
/home/root/whiznium

copy

\$(WHIZDEVROOT): Development root

- init: initialization scripts (machine specific)
- projects: model files and other documentation by project
- rep: repositories - Git / local history by project
- setup: setup scripts, also for new embedded target SDK's
- tools: Java-based WhizniumDBE / SBE Bootstrap & Iterator

\$(WHIZSDKROOT): Software development kit root

- build: source code compilation by project
- lib: dependencies incl. sbe/dbecore and project component libraries

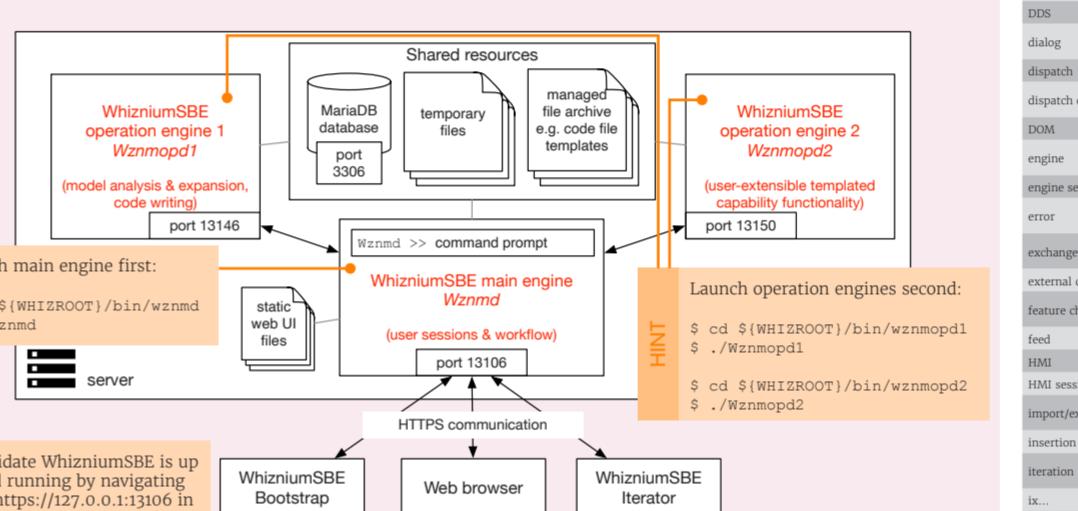
\$(WHIZROOT): Deployment root

- acv: managed file archive by project
- bin: binaries and preferences by project (combined / operation) engine component
- mon: text-based engine monitoring files by project
- tmp: temporary files by project
- web: static web UI files by project

API
application
block
call
capability

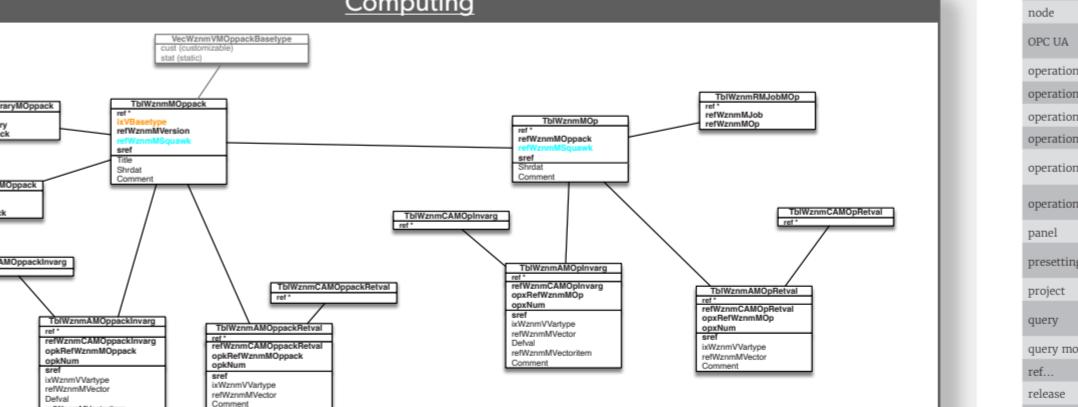
zniumSBE At Runtime

zniumSBE exhibits multi-threaded runtime behavior with functionality



c. multi-line INSERT type	IP cust --- INSERT IP cust --- BEGIN constexpr double pi = 3.141592653589793238462643383279; double radToDeg(double _rad); IP cust --- IEND	b. single-line INSERT type IP include.cust --- INSERT <pre>#include <netcdf.h> // IP include.cust --- LINE</pre>	
d. multi-line LINE replacement	IP handleDownload --- LINE IP handleDownload --- BEGIN (tgzfile.length() > 0) return(xchg(tgmpath + "/" + tgzfile); IP handleDownload --- END	f. KEEP unaltered file IP file --- KEEF	
e. leave header unaltered ABOVE	IP header --- ABOVE IP header --- BEGIN IP header --- END IP header --- LINE	IP header --- ABOVE IP header --- BEGIN IP header --- END IP header --- LINE	
g. KEEP unaltered folder	IP folder --- KEEF	IP folder --- KEEF IP folder --- BEGIN IP folder --- END	

28



Digitized by srujanika@gmail.com

