Programming Languages and Data Access Methods — February 2024

Manufacturer	Programming Language Name	Version	Current Classification	General Availability	Projected	Approved	Divest: Plan	Divest: Execution	Prohibited	Vendor End of Support	References	Notes
General programming language recommendation	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI_SOFTWARE_MEMORY_SAFETY.PDF https://insights.sei.cmu.edu/blog/rust-software-security-a-current-state-assessment/ https://survey.stackoverflow.co/2022/	**EA Direction**: Use of a "memory safe" language is recommended and preferred. Examples of memory safe languages include: C#, Go, Java®, Ruby™, Rust®, Python, and Swift®. Microsoft® revealed at a conference in 2019 that from 2006 to 2018, 70 percent of their vulnerabilities were due to memory safety issues. Google® also found a similar percentage of memory safety vulnerabilities over several years in Chrome®. Malicious cyber actors can exploit these vulnerabilities for remote code execution or other adverse effects, which can often compromise a device and be the first step in large-scale network intrusions. The data bears out, over and over again, that when projects use unsafe languages like C and C++ they are burdened by an avalanche of security vulnerabilities. No matter how talented the engineers, how great the investment in privilege reduction and exploit mitigations, using a language that is not memory safe simply results in too many bugs. These bugs greatly reduce security, as well as stability and productivity. https://www.memorysafety.org/docs/memory-safety/ Commonly used languages, such as C and C++, provide a lot of freedom and flexibility in memory management while relying heavily on the programmer to perform the needed checks on memory references. Simple mistakes can lead to exploitable memory-based vulnerabilities. Software analysis tools can detect many instances of memory management issues and operating environment options can also provide some protection, but inherent protections offered by memory safe software languages can prevent or mitigate most memory management issues. NSA recommends using a memory safe language when possible. While the use of added protections to non-memory safe languages and the use of memory safe languages do not provide absolute protection against exploitable memory issues, they do provide considerable protection. Therefore, the overarching software community across the private sector, academia, and the U.S. Government have begun initiatives to drive the culture of software development towards utilizing memory safe languages. https://endoflife.date/
Oracle	**Java**	25 (LTS)	**Projected**		2026						End of Life Info: https://endoflife.date/java	**EA Direction**: Continue updating to the latest LTS major release. https://en.wikipedia.org/wiki/Java_(programming_language)
		21 (LTS)	Approved	9/19/2023	N	12/31/2023	9/30/2026	9/30/2027	9/30/2028	9/30/2029	End of Life Info: https://endoflife.date/java	https://en.wikipedia.org/wiki/Java_(programming_language) Note: Tracking N and N-1 to LTS versions only while skipping all other version between LTS versions.
		20	**Skipped**									
		19	**Skipped**									
		18	**Skipped**									
		17 (LTS)	Approved	9/14/2021	N-1	12/31/2021	9/30/2025	9/30/2026	9/30/2027	9/30/2029	End of Life Info: https://endoflife.date/java	Java, as developed by the OpenJDK Project, which is owned and primarily employed by Oracle, has been on a 6-month rapid-release cycle since the release of Java 10. Starting with Java 11, new LTS releases occur every six releases, or three years. Java 8 is the last release on the old cycle methodology still in active support. Non-LTS releases are supported for 6 months. The latest supported release in each release cycle can be found at https://www.oracle.com/java/technologies/java-se-glance.html.
		16	**Skipped**	3/16/2021					**Prohibited**	9/30/2021		
		15	**Skipped**	9/16/2020					**Prohibited**	3/31/2021		
		14	**Skipped**	3/17/2020					**Prohibited**	9/30/2020		
		13	**Skipped**	9/17/2019					**Prohibited**	3/31/2021		
		12	**Skipped**	3/19/2019					**Prohibited**	9/30/2019		
		11 (LTS)	Divest Plan	9/25/2018			9/30/2023	9/30/2024	9/30/2025	9/30/2026	End of Life Info: https://endoflife.date/java	
		10	**Skipped**	3/20/2018					**Prohibited**	9/25/2018		
		9	**Skipped**	9/21/2017					**Prohibited**	3/20/2018		
		8 (LTS)	**Prohibited**	3/18/2014					**Prohibited**	3/31/2025		
		7 and earlier	**Prohibited**	7/7/2011					**Prohibited**	7/31/2019		First version GA May 1995 from Sun Microsystems. https://en.wikipedia.org/wiki/Java_(programming_language)
Oracle	**PL/SQL**	All versions	Approved	1995							https://en.wikipedia.org/wiki/PL/SQL	Oracle: https://www.oracletutorial.com/plsql-tutorial/what-is-plsql/ Tech on the Net: https://www.techonthenet.com/oracle/index.php https://endoflife.date/

Programming Languages and Data Access Methods							February 2024					
Manufacturer	**Programming Language Name**	**Version**	**Current Classification**	**General Availability**	**Projected**	**Approved**	**Divest: Plan**	**Divest: Execution**	**Prohibited**	**Vendor End of Support**	**References**	**Notes**
Adobe	**ColdFusion**	2023	Divest Plan	5/31/2023			1/1/2024	6/1/2028	1/1/2029	5/16/2029	Support Date Info: https://helpx.adobe.com/support/programs/eol-matrix.html	**EA Direction**: As applicable update to ColdFusion 2023. Long term - proceed to divest from the ColdFusion programming language entirely prior to it being prohibited in the COV enterprise in 2029. **EA Rationale**: Supporting too many programming languages can create long-term maintenance issues and skill shortage issues within the COV IT enterprise. Licensing, packaging, frameworks, extended ecosystems and support are key considerations in selecting any programming language. **EA goal**: Reduce the quantity of programming languages we officially support. Per Gartner (Dec-2021) and industry programming language trends as evidenced by job postings, the top languages were: Python; Java; JavaScript; C/C++; C#; and PHP. While ColdFusion's CMFL compares to PHP's scripting components, it's tag syntax resembles HTML and it's script syntax resembles JavaScript, EA's position is to divest from this programming language and to not continue investing in it.
		2021	Divest Plan	11/11/2020			1/1/2024	1/1/2026	6/1/2026	11/10/2026	Support Date Info: https://helpx.adobe.com/support/programs/eol-matrix.html	Release graphic: https://endoflife.date/coldfusion Release announcement: https://community.adobe.com/t5/coldfusion-discussions/live-introducing-the-2023-release-of-adobe-coldfusion/td-p/13797706 Downloads: https://helpx.adobe.com/coldfusion/kb/coldfusion-downloads.html#download0 ColdFusion's lifecycle is typically 5 years after release, with new releases usually about every two years. Adobe never announces the release date in advance. https://community.adobe.com/t5/coldfusion-discussions/coldfusion-2023-release-date/td-p/13785784. Adobe release info: https://helpx.adobe.com/coldfusion/kb/coldfusion-downloads.html#downloads1
		2018 and earlier versions	**Prohibited**	7/12/2018						7/13/2024	Support Date Info: https://helpx.adobe.com/support/programs/eol-matrix.html	One of the distinguishing features of ColdFusion is its associated scripting language, ColdFusion Markup Language (CFML). CFML compares to the scripting components of ASP, JSP, and PHP in purpose and features, but its tag syntax more closely resembles HTML, while its script syntax resembles JavaScript. ColdFusion is often used synonymously with CFML, but there are additional CFML application servers besides ColdFusion, and ColdFusion supports programming languages other than CFML, such as server-side Actionscript and embedded scripts that can be written in a JavaScript-like language known as CFScript. Originally a product of Allaire and released on July 2, 1995, ColdFusion was developed by brothers Joseph J. Allaire and Jeremy Allaire. In 2001 Allaire was acquired by Macromedia, which in turn was acquired by Adobe Systems Inc in 2005. ColdFusion is most often used for data-driven websites or intranets, but can also be used to generate remote services such as REST services, WebSockets, SOAP web services or Flash remoting. It is especially well-suited as the server-side technology to the client-side ajax. ColdFusion can also handle asynchronous events such as SMS and instant messaging via its gateway interface, available in ColdFusion MX 7 Enterprise Edition. https://en.wikipedia.org/wiki/Adobe_ColdFusion
CII Honewell Bull	**Ada**	2012	**Prohibited**	2012							https://en.wikipedia.org/wiki/Ada_(programming_language)	https://en.wikipedia.org/wiki/Groupe_Bull https://curlie.org/Computers/Programming/Languages/Ada
Association for Computing Machinery (ACM)	**ALGOL**	DG/L	**Prohibited**	1972							https://en.wikipedia.org/wiki/ALGOL	
IBM	**APL**	APL2	**Prohibited**	1984							https://en.wikipedia.org/wiki/APL_(programming_language)	
IBM	**Assembler**	BAL	**Prohibited**	1964							https://en.wikipedia.org/wiki/IBM_Basic_Assembly_Language_and_successors	
Dartmouth	**BASIC**	All versions	**Prohibited**	1964							https://en.wikipedia.org/wiki/BASIC	
Bell Labs	**"C"**	All versions	**Prohibited**	1973							https://en.wikipedia.org/wiki/The_C_Programming_Language	https://www.section.io/engineering-education/history-of-c-programming-language/
Nantucket Corporation	**Clipper**	xBase	**Prohibited**	1997							https://en.wikipedia.org/wiki/Clipper_(programming_language)	aka CA-Clipper

Programming Languages and Data Access Methods

February 2024

Manufacturer	Programming Language Name	Version	Current Classification	General Availability	Projected	Approved	Divest: Plan	Divest: Execution	Prohibited	Vendor End of Support	References	Notes
CODASYL	**COBOL**	All versions	**Prohibited**	1959							https://en.wikipedia.org/wiki/COBOL	https://en.wikipedia.org/wiki/CODASYL
Embarcadero Technologies	**Delphi**	All versions	**Prohibited**	1995							https://en.wikipedia.org/wiki/Delphi_(software)	https://en.wikipedia.org/wiki/History_of_Delphi_(software)
IBM	**Fortran**	All versions	**Prohibited**	1957							https://en.wikipedia.org/wiki/Fortran	
Software AG	**Natural**		**Prohibited**	1979							https://en.wikipedia.org/wiki/ADABAS#Natural_(4GL)	https://en.wikipedia.org/wiki/Natural_language https://www.xenonstack.com/blog/evolution-of-nlp/
MIT	**Lisp**	All LISP-based languages	**Prohibited**	1958							https://en.wikipedia.org/wiki/Lisp_(programming_language)	https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#List-based_languages_%E2%80%93_LISPs
Unisys / Sperry	**Mapper**	All versions	**Prohibited**	1975							https://en.wikipedia.org/wiki/MAPPER	
Corel	**Paradox**	All versions	**Prohibited**	1985							https://en.wikipedia.org/wiki/Paradox_(database)	
Berkley	**Pascal**	All versions	**Prohibited**	1970							https://en.wikipedia.org/wiki/Pascal_(programming_language)	https://www.britannica.com/technology/Pascal-computer-language
IBM	**PL/I**	All versions	**Prohibited**	1964							https://en.wikipedia.org/wiki/PL/I	aka PL/1
IBM	**Rexx**	All versions	**Prohibited**	1979							https://en.wikipedia.org/wiki/Rexx	
IBM	**RPG X**	All versions	**Prohibited**	1959							https://en.wikipedia.org/wiki/IBM_RPG	

Programming Languages and Data Access Methods

February 2024

Manufacturer	Programming Language Name	Version	Current Classification	General Availability	Projected	Approved	Divest: Plan	Divest: Execution	Prohibited	Vendor End of Support	References	Notes

The following **data access methods** are considered *Prohibited* technologies per Enterprise Technical Architecture (ETA) direction:

https://www.vita.virginia.gov/media/vitavirginiagov/it-governance/ea/pdf/Legacy-IT-Solutions-Topic-Report.pdf

Vendor	Data Access Name	Version	Current Classification	General Availability	Divest: Plan	Divest: Execution	Divest: Plan	Divest: Execution	Prohibited	Vendor End of Support	References	Notes
	Adabas		**Prohibited**								https://documentation.softwareag.com/natural/nat6313win/pg/pg_dbms_ada.htm	https://en.wikipedia.org/wiki/Access_method
	IMS		**Prohibited**								https://flylib.com/books/en/2.869.1.57/1/	https://en.wikipedia.org/wiki/Access_method https://flylib.com/books/en/2.869.1.58/1/ https://en.wikipedia.org/wiki/IBM_Information_Management_System
	VSAM		**Prohibited**								https://en.wikipedia.org/wiki/Virtual_Storage_Access_Method	https://en.wikipedia.org/wiki/Access_method
	ISAM		**Prohibited**								https://en.wikipedia.org/wiki/ISAM	https://en.wikipedia.org/wiki/Access_method
	xBase		**Prohibited**								https://en.wikipedia.org/wiki/Microsoft_Data_Access_Components	https://en.wikipedia.org/wiki/Access_method
	Paradox		**Prohibited**								https://en.wikipedia.org/wiki/Microsoft_Data_Access_Components	https://en.wikipedia.org/wiki/Access_method
	Hierarchical Database Access	All	**Prohibited**								https://en.wikipedia.org/wiki/Access_method	e.g. HSAM, HISAM, HIDAM, HDAM, PHDAM, SHSAM, SHISAM, etc.
	Network Database Access	All	**Prohibited**								https://en.wikipedia.org/wiki/Access_method	e.g. BTAM, QTAM, TCAM, VTAM, etc.
	Non-security patch updated data access methods	All	**Prohibited**								https://en.wikipedia.org/wiki/Access_method	All data access methods and versions without security patching

The following represents COV **programming language classifications** per Enterprise Technical Architecture (ETA):

https://www.vita.virginia.gov/media/vitavirginiagov/it-governance/ea/pdf/Legacy-IT-Solutions-Topic-Report.pdf

Key	Description
	Emerging
	Projected
	Approved
	Divest: Plan
	Divest: Execute
	Prohibited
	Skipped