

TEXTE

71/2020

Guide on Green Public Procurement of Software

TEXTE 71/2020

Ressortforschungsplan of the Federal Ministry for the
Environment, Nature Conservation and Nuclear Safety

Project No. (FKZ) 3715 37 601 0

Report No. (FB) FB000101/ENG

Guide on Green Public Procurement of Software

Recommendations

from UFOPLAN-Project 3715 37 601 0

Entwicklung und Anwendung von Bewertungsgrundlagen
für ressourceneffiziente Software unter Berücksichtigung
bestehender Methodik [Development and application of
criteria for resource-efficient software products with
consideration of existing methods]

by

Jens Gröger


Oeko-Institut e.V., PO Box 17 71, 79017 Freiburg, Germany

On behalf of the German Environment Agency

Imprint

Publisher

Umweltbundesamt
Wörlitzer Platz 1
06844 Dessau-Roßlau
Tel: +49 340-2103-0
Fax: +49 340-2103-2285
buergerservice@uba.de
Internet: www.umweltbundesamt.de

 [umweltbundesamt.de](https://www.facebook.com/umweltbundesamt.de)

 [umweltbundesamt](https://twitter.com/umweltbundesamt)

Report performed by:

Oeko-Institut e.V.
PO Box 17 71
1771 Freiburg
Germany

This guide is based on the following report: Gröger, J.; Naumann, S.; Hilty, L.; Filler, A.; Guldner, A.; Kern, E.; Köhler, A. R.; Maksimov, Y.; Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik [Development and application of criteria for resource-efficient software products with consideration of existing methods]; Oeko-Institut e.V., Hochschule Trier, Environmental Campus Birkenfeld and University of Zurich, Department of Informatics, on behalf of the German Environment Agency, Dessau-Roßlau, September 2018 (UBA-Texte 105/2018).

Report completed in:

December 2018

Edited by:

Advisory Office on Sustainable Information and Communication Technology
Marina Köhn

Publication as pdf:

<http://www.umweltbundesamt.de/publikationen>

ISSN 1862-4804

Dessau-Roßlau, February 2019 (German); June 2020 (English translation)

This guide is based on a study entitled Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik [Development and application of criteria for resource-efficient software products with consideration of existing methods] (UBA 2018).

Despite the thorough examination of all provisions of the guide, error cannot be excluded with absolute security. We therefore assume no guarantee for the correctness, completeness and up-to-dateness of the contents. Any liability of the publisher is also excluded for potential consequences related to the contents.

We allow the copying as well as other forms of usage of all contents of this guide provided that they are not falsified or used in any other improper manner.

Table of content

List of figures	6
List of tables	6
Glossary	7
1 Introduction.....	9
2 How to use this guide	10
2.1 Selecting procurement criteria	10
2.2 Additional information on use of the procurement criteria	13
3 Requirements for sustainable software	14
3.1 Resource efficiency	14
3.1.1 Minimum system requirements	14
3.1.2 Hardware utilization with software in idle mode	15
3.1.3 Hardware utilization and energy consumption during execution of a standard usage scenario	15
3.1.4 Power management support	16
3.2 Anticipated hardware operating life	17
3.2.1 Backward compatibility	17
3.2.2 Cross-platform software and portability	17
3.3 Autonomy of use.....	18
3.3.1 Data format transparency.....	18
3.3.2 Program code transparency.....	18
3.3.3 Continuity of the software product	19
3.3.4 Uninstallability	20
3.3.5 Offline capability	20
3.3.6 Software documentation; licensing and terms of use.....	21
4 Measurement instructions	22
4.1 Reference systems	23
4.2 Standard usage scenarios	24
4.3 Definitions of terms pertaining to measurements and calculations of software properties	26
5 List of references	28

List of figures

Figure 1:	Configuration for measuring the energy and resource efficiency of software ..	22
Figure 2:	Measurement process of hardware capacity load	27

List of tables

Table 1:	Criteria selection based on the type of software being procured	12
Table 2:	Reference systems, 2014 – 2017	24
Table 3:	Overview of the actions carried out for selected software.....	25
Table 4:	Basic definitions.....	26

Glossary

Computing power	For the purposes of this project, the term (maximum) computing power of a central processing unit (CPU) refers to the product of a processor's clock frequency, CPU core count, and width of the data bus. A proportional percentage of available computing power is achieved via any or all of the following: staggering the execution of instructions over time; reducing frequency; not using all of the available processor cores; use of a smaller data bus width.
Content management system	A content management system (CMS) is a software product that is used for content generation by a group of authors, e. g. web content or an in-house company information system.
Evaluation criterion	Requirement to which a product or service is subject and that can be evaluated qualitatively or quantitatively. When comparing products, products with a more favourable assessment will be regarded in a more positive light, e.g. lower cost, lower resource use, longer operating life, and greater usability.
Hardware	The material goods required to run programs or to store or transport data.
Hardware capacity	Quantifiable characteristic of a hardware system which represents its performance limit on a given dimension of performance (e.g., working memory capacity, computing power, bandwidth).
Hardware system	Delimitable unit of hardware that performs defined functions.
Indicator	An empirically determinable quantity that provides insight into a matter that cannot be measured directly. The indicators proposed in this document have different levels of measurement. In some cases, a qualitative ordinal scale is applied (e.g., "insufficient", "sufficient", "good", "very good", or even merely "fulfilled", "not fulfilled"). This is done in order to avoid suggesting a non-existent level of precision through use of a cardinal scale.
Log file	A file automatically created by a program that logs actions performed, including time stamps, while a given measurement such as of a standard usage scenario is being undertaken.
Minimum criterion	Requirements that a given product or service needs to meet in order for the tender to be considered. Failure to meet such requirements will result in the product or service in question being excluded from consideration.
Plugin	A software module that can be integrated into software in order to enhance its functionality.
Reference system	A hardware system that is defined as generally customary in terms of its most important capacities (e.g., working memory, processor performance) during a defined period of time (e.g., one year). The purpose of the reference system is to be able to express indicators such as "minimum local memory" in relation to a reference value (currently "customary" memory).

Resource	In the context of this document, a natural resource, in particular a raw material, a form of energy, or also the capacity of an environmental medium to absorb emissions. To differentiate natural resources from technical ones, especially hardware resources, the more precise term “hardware capacities” is used here for the latter. Since using hardware capacities always results in using natural resources, this distinction (which ultimately amounts to a definitionally difficult differentiation between the ecosphere and the technosphere) is not of decisive importance here.
Resource efficiency	Generally, the amount of “useful work” divided by the amount of resources it requires. In the context of this document, “useful work” is operationalized as the successful execution of standard usage scenarios.
Software	Programs and data in digital form.
Software product / application software	A delimitable unit of programs and data for which a license is available.
Standard configuration	A set of conditions, defined as a reference, under which a given software product is run; it includes the parameter settings selected during installation or operation, the system software provided, potentially additional software products required for operation, as well as the reference system at the hardware level.
Standard usage scenario	A usage scenario that is used for testing a software product and is supposed to be as representative as possible for the customary use case.
System under test (SUT)	A hardware system whose power consumption and hardware capacity use are measured in settings such as during execution of a standard usage scenario. Such a system encompasses both the hardware and software that are required to run the software, for instance the operating system, and runtime environments.
Time stamp	Date and time in a defined digital format that records the exact time that a particular metric was measured or that any other event occurred such as an action within the framework of a standard usage scenario.
Usage pattern	Abstracted form of a sequence of interactions with a given software product.
Usage scenario	Description of a usage pattern which is generally machine executable.
Workload generator	A workload generator simulates work performance on a computer for purposes of software testing. It is usually implemented by automation software or a benchmark program.

1 Introduction

Software has a measurable impact on computer-hardware energy consumption and increasing requirements can make it necessary to replace computer hardware prematurely – a phenomenon known as software obsolescence.

The German Environment Agency (UBA) project “Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik” [Development and application of criteria for resource-efficient software products with consideration of existing methods] (UBA 2018; hereinafter referred to as the “UBA Study”) has developed an evaluation methodology that can be used for energy consumption testing, the utilization of hardware resources and for analysing other environmental characteristics of software products. By comparing various software with the same functions and features, it soon became apparent that in some cases there are substantial differences between the various products. The UBA Study found, for instance, that, depending on use case, the energy consumed during a standard usage scenario can vary from one software product to another by as much as a factor of four. Which means that inefficiently programmed software uses four times more energy than more efficiently designed software. The UBA Study also revealed that when it comes to hardware efficiency (in terms of processor utilization, RAM, non-volatile memory and data transmission), there are substantial differences between software products. This is particularly relevant given the fact that when software places undue strain on computer hardware, computers run slower – prompting both public and private sector actors to replace their purportedly slow machines with newer, faster hardware. Other evaluation criteria that come into play revolve around the user autonomy and user friendliness of software, which will affect the software’s life span.

In the UBA Study a set of 25 criteria and 76 indicators for assessing the environmental impact of software products was developed. For the present guide these were pared down to 13 criteria and 32 sub-criteria that are intended for use in software procurement. This set of criteria facilitates the task of identifying and developing sustainable software. These criteria can be applied not only to commercially available standard software, but also to commissioning the development of software. During the development phase, software products and their efficiency can be continuously improved by repeatedly checking whether the criteria of this guide are being met.

When existing software is being procured, the contracting authority should specifically ask the tenderer whether the characteristics entailed by the criteria of this guide are already implemented, and, when commissioning programming services, should ensure that the contract with the developer mandates that these criteria be met.

This guide to public procurement of sustainable software is addressed to procurement offices. It presents a series of criteria for sustainable software and proposes methods for implementation of these criteria for public tenders. Thus, this guide will hopefully promote the formulation of environmental performance specifications for energy and resource efficient software. Other quality criteria that exceed the scope of environmental requirements and that are not addressed in this guide, include for instance compliance with ISO/IEC 25000 (System and software engineering – systems and software quality requirements and evaluation (SQuaRE)); application of the V-Modell XT software development model promulgated by Germany’s Ministry of the Interior; compliance with regulations concerning computer and software accessibility for the disabled; the security standards promulgated by Germany’s national cyber security authority, the BSI.

2 How to use this guide

2.1 Selecting procurement criteria

Given the fact that software is as varied and multifarious as any other commercially available product, all these procurement guidelines can do is to propose requirements that apply to just about any imaginable software and to adumbrate a basic approach to the procurement of sustainable software. In other words, this, or any other guide, cannot possibly delve into every minute detail and requirement that comes into play in such situations.

Thus, each procurement office or user will need to adapt the criteria of this guide to their own individual situation. Thus, for example, software being procured for use on office computers will have different requirements in terms of maximum levels of computer-system utilization than will be the case for mobile phone software or software for high-performance servers in data centres.

The requirements entailed by the criteria of this guide are couched in terms of evaluation criteria rather than minimum (i.e. exclusion) criteria. This means that software tenderers need to explicitly indicate the extent to which their products meet the criterion. Using this information as a basis, the contracting authority then evaluates the software in question – which will result in more sustainable products receiving a more favourable assessment. Such an assessment can, for example, be based on a numerical scale or a benefit analysis.¹

To this end, the call for tender has to state how such an assessment will be carried out (prioritization or weighting) so that tenderers have the opportunity to offer the product that best meets the requirements.

But there is no getting around the fact that certain criteria need to be flat-out minimum criteria. Which means that software that fails these criteria will be excluded from further consideration and thus from the award procedure. This applies in particular to the first criterion: minimum system requirements. Prior to issuing a call for tender, the contracting authority should determine which hardware and operating system (and which version thereof) are going to be running the software in question. This will enable the contracting authority to determine the minimum requirements (exclusion criteria) so as to ensure that the software will be compatible with existing hardware.

As a rule, the setting of minimum criteria limits the types of software that would be found acceptable. If, for example, one of the requirements is that the software being purchased be open source software, (sub-criterion b, software code transparency), then proprietary software would be found unacceptable. But on the other hand, if the pros and cons of open-source and proprietary software were to be compared, software code transparency should be an assessment criterion, and open software receive a more favourable assessment than proprietary software.

The evaluation criteria for four common types of software are listed in Table 1 below. The range of operating systems, standard software and myriad other types of software that are commercially available today, plus specialized software that is developed under a contract, and optimizations of existing software means that the criteria of this guide can very likely be used for all phases of the procurement process. For instance, when it comes to operating system

¹ For an example of a such an analysis, see “Guide on Green Public Procurement: Products and Services for Data Centres and Server Rooms” issued by the UBA, available at <https://www.umweltbundesamt.de/en/publikationen/guide-on-green-public-procurement-products-services>.

procurement, the criteria can be used to delineate the product being procured – in other words to decide which operating system or version thereof is to be procured. For procurement of standard software – word processing software, spread-sheet software, accounting software, address management software and so on – the criteria can be used to directly compare the pros and cons of a range of tenders. For programming-services procurement, the criteria of this guide readily lend themselves to incorporation into the relevant conditions concerning contract performance: Ideally, the programmers will be contractually obligated to perform their services in a manner that meets all of the criteria related requirements.

Table 1 contains recommendations as to which environmental performance criteria should be used and whether these criteria should be minimum (“M”, in the table below) or evaluation criteria (“E”, in the table below). These two types of criteria can also be combined with each other, so as to formulate minimum requirements plus evaluation of over-fulfilment of minimum criteria. In cases where software development or optimization services are being ordered, certain mandated characteristics of the deliverables can only be checked in the finished software. It is best to formulate such requirements as documentation obligations (“D”, in the table below) in the contractual terms and conditions.

Before determining which criteria are minimum criteria (“M”, in the table below), evaluation criteria (“E”, in the table below) or documentation obligations (“D”, in the table below), it is best to take a very close look at each type of software that is under consideration and, if necessary, adapt the criteria accordingly. It is also advisable to determine whether the requirements can be used as exclusion criteria for their respective sub-criteria – and/or whether these criteria can be used as evaluation criteria.

Table 1: Criteria selection based on the type of software being procured

Criterion	Operating System	Standard Software	Software Development	Optimization of existing software
1 Resource efficiency				
Minimum system requirements	M and E	M and E	M and E	M and E
Hardware usage with software in idle mode	E	E	D	D
Hardware usage and energy consumption during execution of standard usage scenario	-	E	D	D
Power management support	M	M	M	M
2 Anticipated hardware operating life				
Backward compatibility	E	E	D	D
Platform independency and portability	-	-	E	-
3 User autonomy				
Data format transparency	-	M	M	M
Program code transparency	E	E	M or E	M or E
Software continuity	M and E	M and E	M	M
Uninstallability	-	E	M	D
Offline capability	E	E	D	D
Documentation of software, licence and terms of use	E	E	M	M

Legend:

- criterion not applicable

M Minimum criterion

E Evaluation criterion

M and E Exclusion if minimum requirement is not fulfilled plus evaluation of over-fulfilment of minimum criteria

M or E Determination, prior to issuance of the call for tender, whether minimum or evaluation criteria apply

D Contractual obligation to provide documentation after the contract has been performed

2.2 Additional information on use of the procurement criteria

The software assessment methodology proposed here is of relatively recent vintage and is as yet not widely used. Moreover, criteria that require the user to perform measurements on a reference system and for which a standard usage scenario needs to be defined in order to make software comparable with each other, are bound to be challenging for software vendors and developers. Nonetheless, in the interest of avoiding needless waste of energy and resources, it is crucial that contracting authorities implement the process advocated here and require vendors and developers to provide sustainable software.

The ambitious criteria that come into play in this regard are as follows:

- ▶ Hardware capacity load with software in idle mode, and
- ▶ Hardware utilization and energy consumption during execution of a standard usage scenario

Therefore, in order to render the methodology proposed here practicable while still ensuring that the software products in question can be readily compared with each other, the criteria of this guide should be applied pragmatically.

Appendix 3 of the final report of the UBA Study (UBA 2018²) describes how the criteria were applied for the purpose of the study. The procedure described in the said Annex should be factored into the procurement process.

For tips on use of the measurement procedure described in this Guide, see section 4. This section also contains a desktop-computer time series for a reference system and explains how to formulate standard usage scenarios. In addition to the foregoing, a software tool is available from the UBA that was developed during the aforementioned study and that allows for the assessment of software that is tested using log files and that computes the requisite data (see OSCAR 2018; in German).

It is recommended that the advisories in section 4, along with this entire guide, be made available to software vendors who have been asked to submit a bid. For actual calls for tender, it may also make sense either for the contracting authority to provide prospective tenderers with a computer as a reference system that the tenderer can use to test and measure their software's performance, or to have all tests and measurements carried out by an outside test laboratory. In addition, contracting authorities can conduct a pre-tender dialogue with all participating tenderers, so as to ensure a level playing field where all stakeholders have a shared understanding of the mandated criteria – so that tenderers can offer their solutions in a professional, competent and comparable fashion.

² In German; for information in English see KERN et al. 2018

3 Requirements for sustainable software

The requirements described in the following chapter refer to the **Set of Criteria for Sustainable Software**. For more information see list of references, p. 28.

3.1 Resource efficiency

It is essential that the resource use of software functionalities be kept to a minimum. To this end, the resource efficiency of software products should be maximized. For the purposes of this guide, resource efficiency is taken to mean “useful work” divided by the resources being used for such work. The term “useful work” is defined as successful execution of a predefined standard usage scenario (see section 4.2). When it comes to resource efficiency, the whole range of natural resources comes into play, including raw materials, energy, and the ability of the environment to absorb emissions. When it comes to implementing resource efficiency within the framework of this guide, technical and energy resources are used as reference values – especially hardware resources and electricity consumption. Given that hardware capacity use and energy consumption invariably have a draining effect on natural resources, in this guide these phenomena are equated with resource use.

3.1.1 Minimum system requirements

The tender has to give information on the minimum requirements that are essential for the software product to be run.

To this end, the following information has to be given:

- a) Minimum processor architecture requirements (e.g. Intel Core i5, Intel Atom x7)
- b) Minimum local RAM requirements (in MB)
- c) Minimum local permanent memory requirements (in MB)
- d) Minimum display resolution requirements (pixel x pixel)
- e) Minimum internet access bandwidth (Mbit/s)
- f) Requirements for other software (operating system software, middleware, auxiliary applications), e.g. Windows 7, .NET Framework and browser version XY

In vetting various software, preference shall be given to the solutions with the least exacting system requirements.

By way of a minimum criterion for this requirement, additional numerical values shall be mandated for processor architecture, RAM (temporary storage), disk or flash (persistent storage) and the like. These values shall not be exceeded by the minimum requirements of the proposed software. This is a reliable way of ensuring that existing computer hardware will be compatible with the solution being procured, because otherwise it may prove necessary to purchase new hardware for the sole purpose of achieving the desired compatibility.

Requirement	Minimum criterion and evaluation criterion
Proof of compliance	Tenderer documentation

Reference	Set of criteria for sustainable software, criterion 1.1.2
-----------	---

3.1.2 Hardware utilization with software in idle mode

It is essential that software tenders specify the hardware utilization entailed by the software in idle mode. To this end, it is necessary to install the software in a reference system (see section 4.1) and test the hardware utilization as per the measurement procedure described in section 4. Hardware utilization encompasses both the additional utilization resulting from running a given software, plus a percentage of basic utilization (see section 4.3).

To this end, tenderers shall provide the following information:

- Mean processor utilization (in per cent) in idle mode, with a standard configuration
- Mean RAM (temporary storage) use (in MB) in idle mode, with a standard configuration
- Mean disk or flash (persistent storage) use (in MB) in idle mode, with a standard configuration
- Mean bandwidth use (in Mbit/s) for internet access, in idle mode with a standard configuration.

In vetting various products, preference shall be given to those whose hardware resource utilization is lowest.

Requirement	Assessment criterion
Proof of compliance	Measurement log, via performance measurement in a reference system
Reference	Set of criteria for sustainable software, criterion 1.1.3

3.1.3 Hardware utilization and energy consumption during execution of a standard usage scenario

It is essential that the level of hardware use and energy consumption entailed by use of the software in executing a standard usage scenario be specified. To this end, for testing purposes it is essential to install the software in a reference system (see section 4.1) and determine the level of hardware utilization as per the measurement procedure described in section 4.

Hardware utilization is defined as the integral (i.e. summation over time) of hardware usage while a standard usage scenario is being executed. Calculation of this parameter is based on log files that characterize the reference system while the software is being run. As indicated in the set of criteria, hardware utilization encompasses both the additional utilization resulting from running a given software, plus a percentage of basic utilization (see section 4.3).

The metrics for hardware usage is as follows: performance related units such as %*s (processor work); MByte*s (RAM work); MByte/s*s = MByte disk or flash (persistent storage work, reading and writing); and MBit/s*s = MBit (amount of data transferred in the network).

In addition to reference-system hardware use, its electricity consumption also needs to be measured. This characteristic constitutes the integral of electricity consumption over time (while the relevant standard usage scenario is being executed). Contrary to the aforementioned

observations concerning hardware utilization, when it comes to measuring electricity consumption only the power exceeding basic reference-system energy consumption (net power) is taken into account.

The following metrics need to be included under “standard configuration” in each standard usage scenario:

- a) Processor work (%*s)
- b) RAM (temporary storage) work (MByte*s)
- c) Disk or flash (persistent storage) work (read/write) (MByte/s*s)
- d) Internet access bandwidth (Mbit*s)
- e) Net energy consumption (Wh)

In vetting various software products, preference shall be given to those whose hardware use and energy consumption are lowest.

Requirement	Assessment criterion
Proof of compliance	Measurement procedure, via performance measurement in a reference system
Reference	Set of criteria for sustainable software, criteria 1.1.4 and 1.2

3.1.4 Power management support

It is crucial that software does not prevent the computer from reducing the hardware utilization and thus energy consumption, when in hibernation mode or stand-by. The power management of the computer or its associated client systems must not be compromised. Also, the software functionality must not be impaired by the power management, e.g. via data loss or impaired operability. Neither the need for (data loss-free) repeat login to server-based software nor the time it takes to reactivate a computer system that is in power saving mode is regarded as constituting impaired operability or functionality.

The following requirement needs to be met:

- a) Even when power management is activated for the underlying layers of a computer system or of the associated client systems, the software remains fully functional. For server-based software, this means that activating client-side power management does not impair functionality (e.g. no loss of session information if the client switches to hibernation mode).

If a tender fails this criterion, it shall be excluded from further consideration.

Requirement	Minimum criterion
Proof of compliance	Tenderer declaration
Reference	Set of criteria for sustainable software, criterion 1.3.2

3.2 Anticipated hardware operating life

Neither off-the-shelf nor commissioned software should result in a situation where existing hardware needs to be taken out of service because it is incompatible with the new software. By the same token, software updates should not necessitate hardware updating. Instead, it should be possible for users to manage the replacement of software and hardware separately. Thus, the requirements laid out in this section address the potential operating life of computer hardware.

3.2.1 Backward compatibility

Tenderers of software products shall be obligated to submit a binding declaration indicating which reference system from which year the software in question will be compatible with. The relevant reference systems can be found in section 4.1. In determining said year, the requisite ancillary software (operating system, framework, application software) needs to be taken into account.

In such cases, tenderers shall provide the following information:

- a) Year of release of the reference system (calendar year)

In vetting various software, preference shall be given to those that are compatible with older reference systems.

Requirement	Assessment criterion
Proof of compliance	Tenderer documentation
Reference	Set of criteria for sustainable software, criterion 2.1

3.2.2 Cross-platform software and portability

Ideally, a given software should be compatible with a range of commonly used hardware and software system environments. In switching from one system environment to another, software should remain fully operational and users should not encounter any difficulties in this regard.

Tenderers shall provide the following information:

- a) Operating systems and the versions thereof (and if applicable, any runtime environments) with which the software in question is compatible.

In vetting various software, preference shall be given to those that are compatible with a range of productive system environments. In making such determinations, it is vital that the system environments currently in use (or intended for future use) in the institution in question be duly taken into consideration.

Requirement	Assessment criterion
Proof of compliance	Tenderer documentation

Reference	Set of criteria for sustainable software, criterion 2.2
-----------	---

3.3 Autonomy of use

Ideally, software should allow users maximum freedom to determine for themselves how they will use the software. In other words, the software should afford users maximum freedom of choice as to product life span and hardware capacity use (e.g. so that users are free to install only the features they need) – and thus in respect to natural resources, insofar as possible. The requirements listed below help to achieve various aspects of user autonomy.

3.3.1 Data format transparency

To ensure the interoperability of the procured software product with software that is currently in use or might be procured at a later date, the data formats (i.e. file and stream formats) that the software uses to store user-generated data or to exchange data with other programs needs to be adequately documented. The data formats in question need to comply with open standards, enabling further use of the data with another software product.

In order for this criterion to be met, contracting authorities need to indicate the following for sub criterion b), which standards will count as open standards at the time when the contract is awarded and/or how the openness of the standard can be verified.³

Thus, tenderers shall be required to provide the following materials and information:

- a) The manuals or technical data sheets that document the data formats.
- b) Indication of the data formats and their assignment to an open standard.
- c) Examples of other software products (sourced from other vendors) that can process these data formats.

If a proffered solution fails this criterion, it shall be excluded from further consideration.

Requirement	Minimum criterion
Proof of compliance	Tenderer documentation
Reference	Set of criteria for sustainable software, criterion 3.1.1

3.3.2 Program code transparency

If you intend to add new functionalities to the software you are purchasing, it is vital that the APIs (application programming interfaces) be clearly documented, and that the API interfaces comply with open standards.

If you also intend to optimize the software without support from the software tenderer, it is vital that the tenderer discloses the source code in whole or in part; and that you obtain the

³ Also see European Interoperability Framework, available at <http://ec.europa.eu/idabc/en/document/3473/5585.html>

tenderer's permission to modify the software. To do this, the software should be released under a license that applies to your specific use case.

Thus, tenderers shall be required to provide the following materials and information:

- a) If APIs exist: Documentation of the interfaces,
- b) Information on the extent to which program source code will be disclosed for the various components of the software; or whether the source code for the entire software product will be disclosed.
- c) Indication of the license under which the software product is released, and whether it allows third parties to further develop the software.

You will need to determine which, if any, of the aforementioned criteria to apply, how to weight them, depending on your particular situation; or whether or not to make them exclusion criteria (i.e. minimum requirements).

In vetting various software, preference shall be given to those that exhibit a high degree of transparency and interoperability.

Requirement	Minimum criterion and assessment criterion
Proof of compliance	Tenderer documentation
Reference	Set of criteria for sustainable software, criterion 3.1.2

3.3.3 Continuity of the software product

It should be possible to use newly acquired software over a relatively long period of time without running into any serious IT security or other problems. The software you decide to purchase should give you the option to choose whether to install security updates only, or other functionality or other types of updates as well.

Thus, tenderers shall be required to provide the following information:

- a) The period during which the tenderer will guarantee that security updates will be provided for the software in question.
- b) Lead time (number of days) for providing security updates in the event a security vulnerability comes to light.
- c) Indication as to whether security updates will be differentiated from other updates and whether it will be possible for users to forego non-security updates.

For such criteria, it is vital that you define minimum requirements as minimum (exclusion) criteria (minimum lead time for security updates, maximum response time) and any over-fulfilment of the minimum requirements should be positively taken into account. In evaluating tenders using evaluation criteria as a basis, a more positive assessment shall be granted to those tenderers whose software allows for greater continuity.

Requirement	Minimum criterion and assessment criterion
Proof of compliance	Tenderer documentation
Reference	Set of criteria for sustainable software, criterion 3.1.3

3.3.4 Uninstallability

It should be possible to uninstall decommissioned software in such a way that no unnecessary traces of it remain in the computer system. This of course does not include any data that was output and/or processed using the software – which means that such data should not be deleted automatically when the software itself is deleted from your system. To this end, it should be possible for users and/or system administrators to delete the software quickly and easily, including any components or libraries installed by the software itself.

Thus, tenderers shall be required to provide the following information:

- a) Description of the software uninstallation procedure.
- b) The number of files and folders, the amount of data, and any registry entries or the like that will remain in the system after uninstallation.

In vetting various software, preference shall be given to those that are the easiest to uninstall and that leave the fewest traces in the system.

Requirement	Minimum criterion or assessment criterion
Proof of compliance	Measurement log for a reference system. If no reference system is defined, these measurements can be carried out on a computer system specified by the tenderer.
Reference	Set of criteria for sustainable software, criterion 3.2.1

3.3.5 Offline capability

The functionality and/or availability of the software should not be compromised by external factors such as license server availability. Moreover, the software product should be largely functional offline – unless of course the network connectivity is explicitly necessary for providing the functionality.

Thus, tenderers shall be required to provide the following:

- a) An assessment of offline capacity, using the following classifications:
 - a. Offline use possible
 - b. Offline use possible with limitations (the limits shall be specified)
 - c. Offline use impossible

In vetting various software, preference shall be given to those that offer a high degree of offline operability.

Requirement	Assessment criterion
Proof of compliance	Tenderer documentation
Reference	Set of criteria for sustainable software, criterion 3.4.1

3.3.6 Software documentation; licensing and terms of use

The software product has to include documentation that ensures that the software can be used on a long term and in a resource-efficient fashion. In particular, installation, uninstallation, and data importing and export should be explained, along with ways to reduce resource use (e.g. activating power saving modes, deactivating unused modules, controlling the size of swap files, expunging traces of deleted data). This documentation should also contain information on license and usage conditions so as to ensure the legality of any further development of the software. The documentation can be integrated into the software as a separate text file or in a help system.

Tenderers shall be required to provide the following:

a) Software documentation

In vetting various software, preference shall be given to those whose documentation is the clearest and most user friendly.

In the case of commissioning software development, submission of the software documentation shall be a contractual requirement.

Requirement	Minimum criterion or assessment criterion
Proof of compliance	Submission of product documentation
Reference	Set of criteria for sustainable software, criterion 3.5.1

4 Measurement instructions

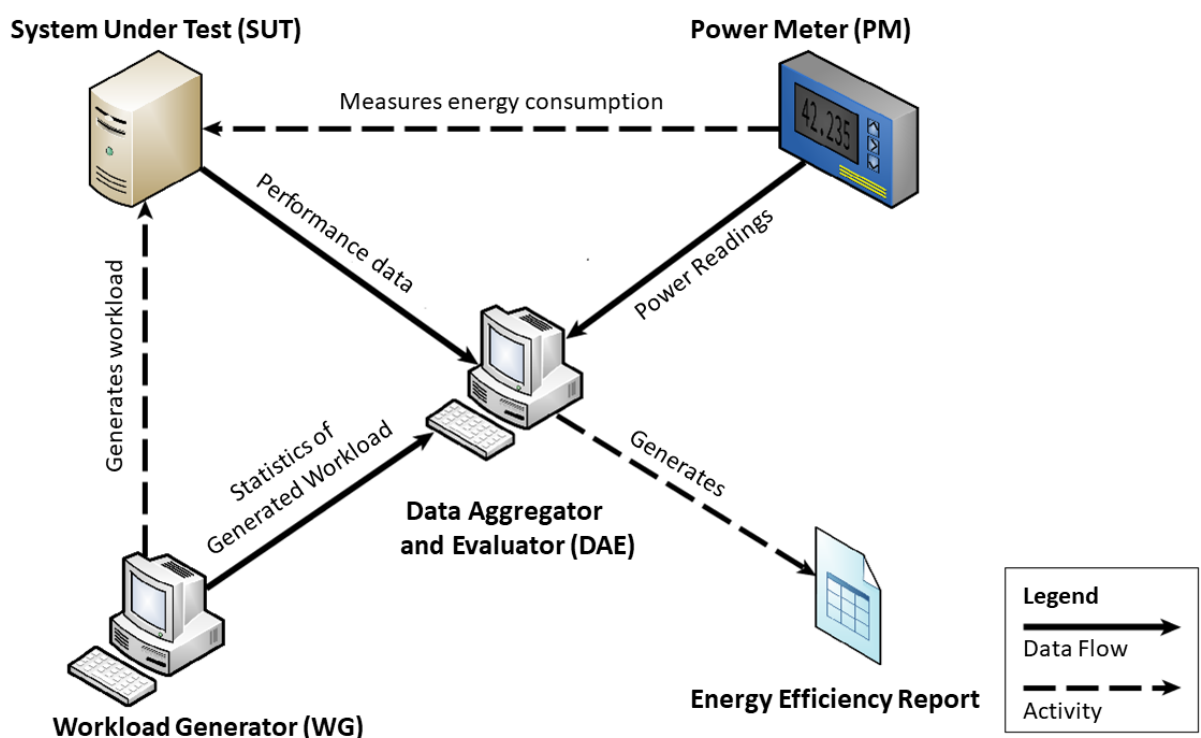
A precondition for meeting some of the aforementioned requirements is submitting a reference-system measurement report in accordance with the measurement procedures set forth in this document. This is the only viable way to compare the information provided by various tenderers.

This pertains to the following requirements:

- ▶ Hardware utilization with the software in idle mode
- ▶ Hardware utilization and energy consumption during execution of a standard usage scenario
- ▶ Backward compatibility
- ▶ Uninstallability

Two requirements need to be met in order for the requisite measurements to be performed. The software needs to be installed and run on a System Under Test (SUT). The reference system needs to be equipped with suitable energy measurement devices, and hardware utilization must be documented via a data logger. It is also necessary to define a standard usage scenario that corresponds to typical use of the software, thus enabling a replicable sequence of commands and user interactions as generated load. The graphic below illustrates the measurement configuration.

Figure 1: Configuration for measuring the energy and resource efficiency of software



Source: UBA 2018

In the following sections, rules for defining a reference system and standard usage scenarios are presented, as described in the UBA Study (UBA 2018). When issuing calls for tender or carrying

out market research, you need to specify the relevant reference system and standard usage scenario.

The measurements shall be performed either by a suitable test lab, or by the tenderer, if they have the requisite technical capability at their disposal. Software for this process is available from the UBA (OSCAR 2018, in German). This evaluation software analyses the reference system's hardware utilization and energy consumption log files and contains complete instructions for the measurements of energy consumption and the utilization of hardware, using standard benchmarking tools. For further information on criteria use, also see Annex 3 of the research report (UBA 2018, in German) that describes the operationalization of indicators.

4.1 Reference systems

The tender specifications that you submit to bidders need to specify a reference computer system that corresponds to the actual usage scenarios that come into play in your particular situation. For example, for software that is intended for local installation in a government office, a reference system would consist of a desktop computer with the type of standard hardware commonly used in government offices and a commonly used operating system. On the other hand, if the software in question is intended for use in a server application or is being commissioned or purchased for a smartphone or the like, you will need to find a different – and suitable – computer system as a reference system.

For the following measurement tasks a reference system needs to be specified:

- ▶ Hardware usage with software in idle mode
- ▶ Hardware usage and energy consumption for a standard usage scenario
- ▶ Uninstallability

In the event you choose not to specify a reference system, the requisite measurements for uninstallability can be performed on an alternative computer system.

In the aforementioned study (UBA 2018), the reference systems were the computer systems that are described in Table 2.⁴ For an actual call for tender, you might, for example, want to specify a 2017 reference system, or an older system (e.g. from 2014) if your hardware is outdated. For future tenders, you can of course also select newer computer systems with more current specifications. The UBA Study (UBA 2018) also refers to older reference systems (from 2010 onwards) that are particularly suitable for assessing:

- ▶ backward compatibility.

⁴ They characterize the range of computer equipment commonly found in government offices.

Table 2: Reference systems, 2014 – 2017

Technical parameter	2014 – 2015	2016	2017
Manufacturer	Fujitsu	Fujitsu	Fujitsu
Model	Esprimo P920 ⁵	Esprimo P956 ⁶	Esprimo P957 ⁷
Processor	Intel i5-4590	Intel i5-6500	Intel i5-6500
Number of cores	4	4	4
Clock rate	3,3 GHz	3,2 GHz	3,2 GHz
RAM	4 GB (DDR4, 2133 MHz)	4 GB (DDR4, 2133 MHz)	8 GB (DDR4, 2133 MHz)
Hard drive (HDD / SSD)	HDD SATA III 500 GB (6 Gbit/s)	HDD SATA III 500 GB (6 Gbit/s)	SSD SATA III 128 GB (6 Gbit/s)
Graphics card	Intel HD Graphics 4600	Intel HD Graphics 530	Intel HD Graphics 530
Network	LAN GigaBit	LAN GigaBit	LAN GigaBit
Operating system	Win 8.1	Win 8.1	Win 10
Efficiency of power supply	91% - 94%	84% - 92%	84% - 92%

Source: UBA 2018

4.2 Standard usage scenarios

The standard usage scenario simulates what occurs during actual use of software. Such scenarios include the execution of tasks for which the software was developed or purchased, and, if necessary, interactions with the software user. Using uniform standard usage scenarios will enable you to, for example, compare the energy consumption and hardware utilization of software of the same type.

The development of standard usage scenarios should be subject to a transparent process involving all stakeholders – mainly potential software users, as well as software developers. In the case of an actual call for tender, the contracting authority might specify the standard usage scenario.

In the interest of ensuring that various software of the same type can be readily compared with each other, the same standard usage scenario should be executed for each such software. The following factors are relevant for the development of standard usage scenarios:

- Standard usage scenarios are usually generated with automation software such as WinAutomation. The scenario contains standard tasks that are carried out with the software. In developing such scenarios, care should be taken to ensure that (a) waiting time that simulates an actual user is inserted prior to each action; and (b) the maximum possible

⁵ Data sheet (German): <https://sp.ts.fujitsu.com/dmsp/Publications/public/ds-ESPRIMO-P920-0Watt-de.pdf>

⁶ Data sheet (German):: <https://sp.ts.fujitsu.com/dmsp/Publications/public/ds-ESPRIMO-P956-E90-de.pdf>

⁷ Data sheet (German):: <https://sp.ts.fujitsu.com/dmsp/Publications/public/ds-ESPRIMO-P957-E90-de.pdf>

number of different standard user actions is carried out that arise during typical work processes.

- ▶ In developing standard usage scenarios, it is essential that the test procedures are replicable for all of the software products that are being compared with each other. To this end, before a scenario is created, you will need to verify that all of the actions entailed by it also occur in the software products that are being compared with each other; and ideally these actions should be carried out with standard configurations and without additional packages or plugins.
- ▶ Make sure that when the same test is run multiple (e.g. 30) times, in the interest of achieving comparable test runs, all changes that occur in the reference system across each iteration of the same test are reversed before the next test is run. Any system processes such as virus scanners that run intermittently and that skew the test results need to be deactivated.
- ▶ Use a time stamp to synchronize your tests. Hence all hardware utilization and power measuring points need to be time stamped. The automation software also generates and saves a log file indicating when each test run began and ended, as well as a time stamp indicating when each action that was executed began and ended. This makes it possible to determine the exact extent of hardware utilization and energy consumption for each individual action.
- ▶ In order to enable processing and subsequent replication, measurements, log files and automation scripts are stored in a central location.

Table 3 contains a summary of the actions carried out within a standard usage scenario of the type used in the UBA study (UBA 2018).

Table 3: Overview of the actions carried out for selected software

Word processing	Browser
Entire text processed	E-mails read and written
Table of contents inserted and updated	Web video stream viewed
View adjusted	Online store visited
Content inserted and edited	Bookmarks set
PDF file generated	Add-on installed
Data saved	File downloaded
Content Management Systems	Databases
Responses to comments composed	Scheme already available
New page created	Data entered
All pages published	Data read
PDF files uploaded	Data modified
PDF files linked	Data deleted
Page viewed	230 runs per function
Additionally: Workload generated for simulation of visitors	120,000 accesses per run

Source: UBA 2018

4.3 Definitions of terms pertaining to measurements and calculations of software properties

Table 4: Basic definitions

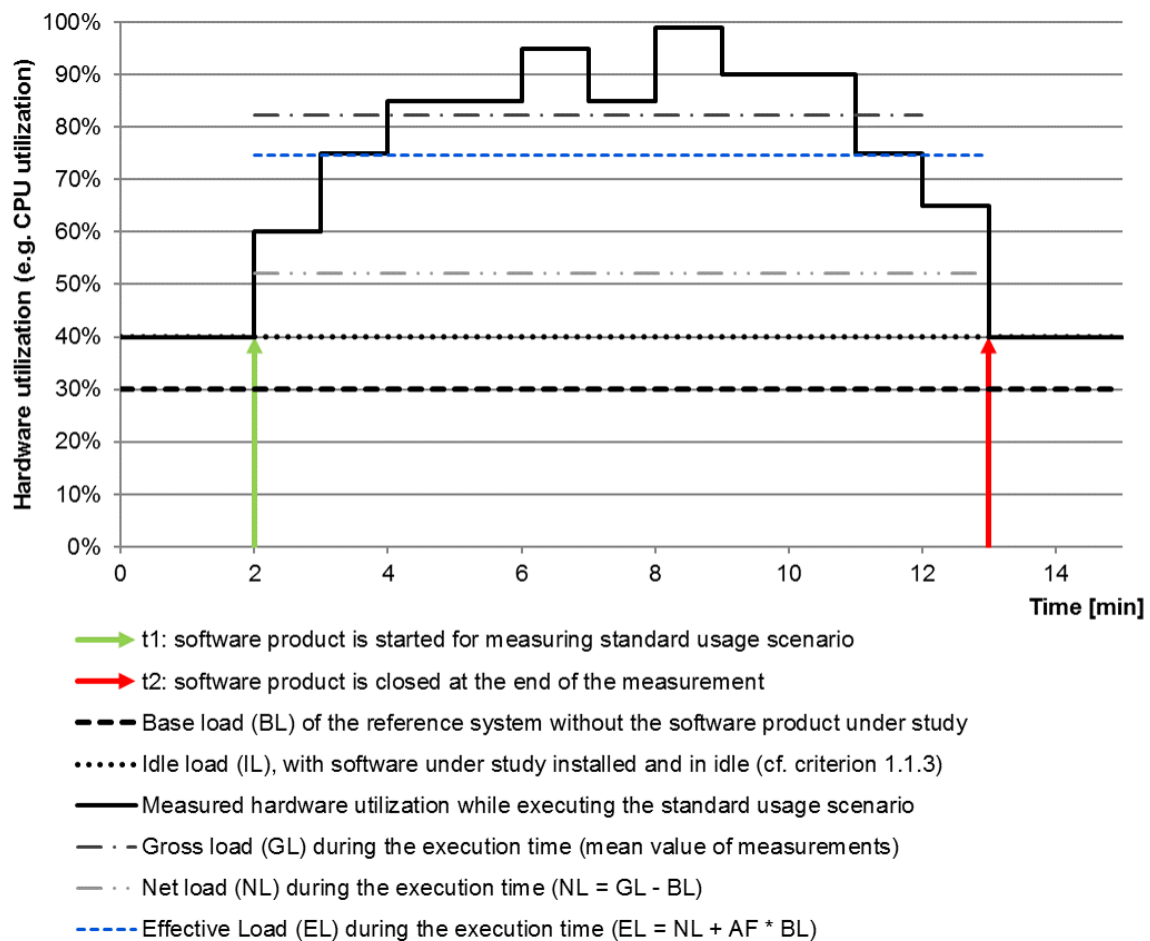
Identifier	Name	Definition	Comments
FL_i	Full load	Upper limit of the capacity i in the reference system.	For processing power, the FL is 100%, for working memory the sum of the installed RAM, for network bandwidth the maximum transmission speed, etc.
BL_i	base load	Average load of the capacity i in the reference system when the software product under study is not installed	
IL_i	idle load	Average load of the capacity i in the reference system when the software product under study is installed, but idle.	Idle load includes base load.
NIL_i	net idle load	$NIL_i = IL_i - BL_i$	
t	time	Time needed to execute the standard usage scenario on the reference system.	Begins with the start of the standard usage scenario and ends when all required actions are executed, including follow-up processes (such as releasing memory or deleting temporary files).
GL_i	gross load	Load of the capacity i in the reference system while executing the standard usage scenario, measured as time-weighted average over t .	
NL_i	net load	$NL_i = GL_i - BL_i$	
AF_i	Allocation factor	$AF_i = NL_i / (FL_i - BL_i)$	Allocation factor used to assign a share of the base load GA to the effective load EL (defined below).
AFI_i	allocation factor idle	$AFI_i = NIL_i / (FL_i - BL_i)$	Allocation factor used to assign a share of the base load GA to the effective load idle ELI (defined below).

Identifier	Name	Definition	Comments
EL _i	effective load	$EL_i = NL_i + AF_i * BL_i$	
EIL _i	effective load idle	$EIL_i = NIL_i + AFI_i * BL_i$	Used to calculate the indicators for hardware demand of criterion 1.1.3
HD _i	hardware demand	$HD_i = EL_i * t$	Used to calculate the indicators for hardware demand of criterion 1.1.4

Source: UBA 2018

The figure below shows a measurement cycle, along with the attendant loads and their respective designations.

Figure 2: Measurement process of hardware capacity load



Source: UBA 2018

5 List of references

OSCAR 2018:

Open Source Software Consumption Analysis; Auswertungssoftware zu Hardwareauslastung und Energieverbrauch; Birkenfeld; Dezember 2018;
<https://oscar.umwelt-campus.de/> (in German)

Set of criteria:

Hilty, L.; Naumann, S.; Maksimov, Y.; Kern, E.; Filler, A.; Guldner, A.; Gröger, J.; Kriterienkatalog für nachhaltige Software / Set of Criteria for Sustainable Software

English: <https://www.umwelt-campus.de/en/research/projekte/green-software-engineering/set-of-criteria/introduction>; German: <https://www.umwelt-campus.de/forschung/projekte/green-software-engineering/kriterienkatalog>; including PDF-

files of both the English and German version as of May 2017.

Version May 2017 in German also in: UBA 2018 (Appendix 1, page 92 ff);

UBA 2018:

Gröger, J.; Naumann, S.; Hilty, L.; Filler, A.; Guldner, A.; Kern, E.; Köhler, A. R.; Maksimov, Y.; Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik [Development and application of criteria for resource-efficient software products with consideration of existing methods]; Öko-Institut e.V. in Kooperation mit Hochschule Trier, Umwelt-Campus Birkenfeld und Universität Zürich, Institut für Informatik im Auftrag des Umweltbundesamtes; Dessau-Roßlau; Dezember 2018;
<https://www.umweltbundesamt.de/publikationen/entwicklung-anwendung-von-bewertungsgrundlagen-fuer> (in German)

For further information in English please see also:

KERN et al 2018:

Kern, E.; Hilty, L.; Guldner, A.; Maksimov, Y.; Filler, A.; Gröger, J.; Naumann, S. (2018): Sustainable software products—Towards assessment criteria for resource and energy efficiency. In: Future Generation Computer Systems 86 (2018) 199–210; April 2018.

<https://doi.org/10.1016/j.future.2018.02.044>

The German version of this Guide is available at

<https://www.umweltbundesamt.de/publikationen/leitfaden-zur-umweltfreundlichen-oeffentlichen-21>