ITUPublications
Recommendations

International Telecommunication Union
Standardization Sector

Recommendation

ITU-T Y.3532 (05/2023)

SERIES Y: Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities

Cloud Computing

Cloud computing – Functional requirements of Platform as a Service for cloud native applications

ITU-T Y-SERIES RECOMMENDATIONS

Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities

GLOBAL INFORMATION INFRASTRUCTURE	Y.100-Y.999
General	Y.100-Y.199
Services, applications and middleware	Y.200-Y.299
Network aspects	Y.300-Y.399
Interfaces and protocols	Y.400-Y.499
Numbering, addressing and naming	Y.500-Y.599
Operation, administration and maintenance	Y.600-Y.699
Security	Y.700-Y.799
Performances	Y.800-Y.899
INTERNET PROTOCOL ASPECTS	Y.1000-Y.1999
General	Y.1000-Y.1099
Services and applications	Y.1100-Y.1199
Architecture, access, network capabilities and resource management	Y.1200-Y.1299
Transport	Y.1300-Y.1399
Interworking	Y.1400-Y.1499
Quality of service and network performance	Y.1500-Y.1599
Signalling	Y.1600-Y.1699
Operation, administration and maintenance	Y.1700-Y.1799
Charging	Y.1800-Y.1899
IPTV over NGN	Y.1900-Y.1999
NEXT GENERATION NETWORKS	Y.2000-Y.2999
Frameworks and functional architecture models	Y.2000-Y.2099
Quality of Service and performance	Y.2100-Y.2199
Service aspects: Service capabilities and service architecture	Y.2200-Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250-Y.2299
Enhancements to NGN	Y.2300-Y.2399
Network management	Y.2400-Y.2499
Computing power networks	Y.2500-Y.2599
Packet-based Networks	Y.2600-Y.2699
Security	Y.2700-Y.2799
Generalized mobility	Y.2800-Y.2899
Carrier grade open environment	Y.2900-Y.2999
FUTURE NETWORKS	Y.3000-Y.3499
CLOUD COMPUTING	**Y.3500-Y.3599**
BIG DATA	Y.3600-Y.3799
QUANTUM KEY DISTRIBUTION NETWORKS	Y.3800-Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	Y.4000-Y.4999
General	Y.4000-Y.4049
Definitions and terminologies	Y.4050-Y.4099
Requirements and use cases	Y.4100-Y.4249
Infrastructure, connectivity and networks	Y.4250-Y.4399
Frameworks, architectures and protocols	Y.4400-Y.4549
Services, applications, computation and data processing	Y.4550-Y.4699
Management, control and performance	Y.4700-Y.4799
Identification and security	Y.4800-Y.4899
Evaluation and assessment	Y.4900-Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.3532

Cloud computing – Functional requirements of Platform as a Service for cloud native applications

Summary

Recommendation ITU-T Y.3532 provides overview and functional requirements of Platform as a Service (PaaS) for cloud native applications. To introduce cloud native PaaS, this Recommendation also provides an overview of cloud native and cloud native applications. This Recommendation also addresses functional requirements of PaaS for cloud native applications through various use cases.

History *

Edition	Recommendation	Approval	Study Group	Unique ID
1.0	ITU-T Y.3532	2023-05-14	13	11.1002/1000/15537

Keywords

Cloud native, cloud native application, functional requirements, platform as a service, PaaS.

* To access the Recommendation, type the URL https://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T Y.3532

Cloud computing – Functional requirements of Platform as a Service for cloud native applications

1 Scope

This Recommendation provides an overview of Platform as a Service (PaaS), cloud native and its relationship. The functional requirements of PaaS for cloud native applications are provided also from use cases. It addresses the following subjects:

- Overview of PaaS;
- Overview of cloud native;
- Relationship between PaaS and cloud native applications;
- Functional requirements of PaaS for cloud native applications;
- Typical use cases to derive functional requirements.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T X.1601] Recommendation ITU-T X.1601 (2015), *Security framework for cloud computing.*

[ITU-T Y.3500] Recommendation ITU-T Y.3500 (2014), *Information technology – Cloud computing – Overview and vocabulary.*

[ITU-T Y.3502] Recommendation ITU-T Y.3502 (2014), *Information technology – Cloud computing – Reference architecture.*

[ITU-T Y.3535] Recommendation ITU-T Y.3535 (2022), *Cloud computing – Functional requirements for a container.*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 cloud native application [b-ISO/IEC TS 23167]: A type of cloud application that is explicitly designed to run within and to take advantage of capabilities and environment of cloud services.

3.1.2 cloud service [ITU-T Y.3500]: One or more capabilities offered via cloud computing invoked using a defined interface.

3.1.3 cloud service category [ITU-T Y.3500]: Group of cloud services that possess some common set of qualities.

3.1.4 cloud service customer (CSC) [ITU-T Y.3500]: A person or organization that consumes delivered cloud services within a contract with a cloud service provider.

3.1.5 cloud service provider (CSP) [ITU-T Y.3500]: An organization that provides and maintains delivered cloud services.

3.1.6 container [ITU-T Y.3535]: A set of software to provide isolation, resource control and portability for virtualization processing of an application.

NOTE 1 – Container runs on the kernel in a bare-metal machine or virtual machine.

NOTE 2 – "Application" implies business logic including a required library or binary to run in a container.

3.1.7 execution environment [b-ITU-T Y.4500.1]: Logical entity that represents an environment capable of running software modules.

3.1.8 infrastructure as a service (IaaS) [ITU-T Y.3500]: Cloud service category in which the cloud capabilities type provided to the cloud service customer is an infrastructure capabilities type.

3.1.9 middleware [b-ITU-T Y.101]: The mediating entity between two information elements. Such an element can be, for example, an application, infrastructure component or another mediating entity.

3.1.10 platform as a service (PaaS) [ITU-T Y.3500]: Cloud service category in which the cloud capabilities type provided to the cloud service customer is a platform capabilities type.

3.1.11 platform capabilities type [ITU-T Y.3500]: Cloud capabilities type in which the cloud service customer can deploy, manage and run customer-created or customer-acquired applications using one or more programming languages and one or more execution environments supported by the cloud service provider.

3.1.12 programming language [b-ISO/IEC/IEEE 24765]: Language used to express computer programs.

3.1.13 runtime environment [b-ISO/IEC TR 13066]: A software environment that provides all of the resources necessary for software applications to run, yet is not itself an operating system

3.1.14 software as a service (SaaS) [ITU-T Y.3500]: Cloud service category in which the cloud capabilities type provided to the cloud service customer is an application capabilities type.

3.2 Terms defined in this Recommendation

This Recommendation defines the following term:

3.2.1 cloud native: A methodology of creating software, which takes full advantage of capabilities of cloud computing during the process of software design, development, deployment, running and management.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API Application Programming Interface

CPU Central Processing Unit

CSC Cloud Service Customer

CSP Cloud Service Provider

IaaS Infrastructure as a Service

IDE Integrated Development Environment

PaaS Platform as a Service

5 Conventions

In this Recommendation:

The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.

6 Overview of PaaS

6.1 Service object of PaaS

When a cloud service customer (CSC) creates an application in cloud computing, which is also known as a CSC developing software in cloud computing, the CSC completes four phases: application coding, application testing, application deployment and application maintenance.

– **Application coding:** the CSC develops software by writing code using a chosen programming language.

– **Application testing:** the CSC tests the developed application to verify the entire application works as expected. Within this phase, the CSC keeps finding and fixing bugs until achieving a bug-free, working and stable software.

– **Application deployment:** the CSC deploys the tested application into a chosen execution environment. After application deployment, the CSC runs the application.

– **Application maintenance:** the CSC continuously manages the running application, periodically maintaining the application by fixing bugs and upgrading software based on user feedbacks.

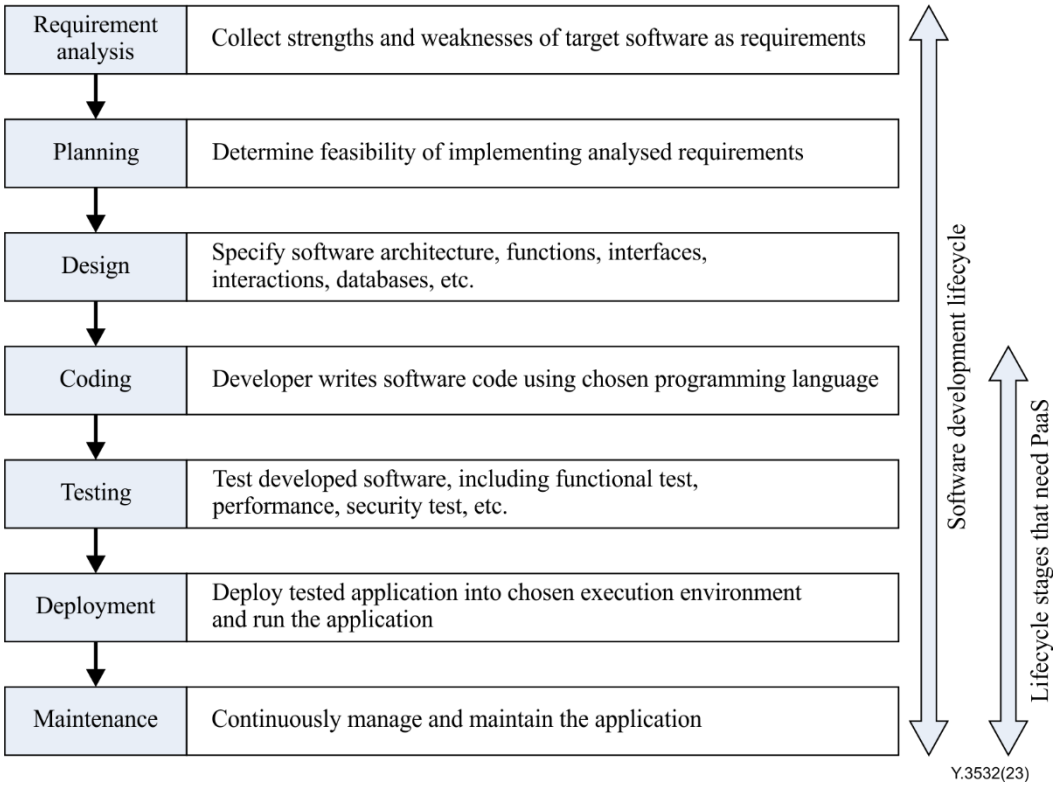


Figure 6-1 – Example of application development lifecycle and relationship with PaaS

To support the CSC's creating an application in cloud computing, the cloud service provider (CSP) provides a group of cloud services to the CSC for application coding, testing, deployment and maintenance. The cloud service category of these group of cloud services is platform as a service (PaaS).

6.2 Introduction to PaaS

According to [ITU-T Y.3500], PaaS is a group of cloud services in which the capability provided to the CSC can support them to deploy, manage and run cloud applications using one or more programming languages and one or more execution environments supported by the CSP.

PaaS contains any of the three features identified in [b-ITU-T Y.3501] as follows:

– PaaS provides **application hosting environment** on which cloud applications are rapidly deployed, stably executed, flexibly managed and isolated from other applications.

  Within the application hosting environment, the CSP provides the CSC with execution environments containing the application's required physical and virtualized resources (computing, storage and network), operating systems, runtime environment, middleware, application dependent software, etc. Besides, as the CSC creates or acquires cloud applications using one or multiple programming languages, the CSP provides matched execution environments for applications developed in different programming languages.

  NOTE 1 – Application dependent software is a type of software that provides application implementation and running necessities. For example, database, router, firewall and load balancer.

  The CSP also reduces the complexity for the CSC to manage a cloud application by taking over the installation, configuration and management of the application-hosting environment and leaving only the cloud application code for the CSC to provide. This constructs a "push and run" environment for cloud applications.

– PaaS provides an **integrated development environment** (IDE) as well as **development tools** to support the CSC to achieve application coding, testing and maintenance in cloud computing.

  Development tools usually include a code editor, code repositories, build tools, debug tools, test tools, security tools, monitoring services and analytics services.

  An IDE is software combining a group of development tools and providing the CSC with comprehensive facilities for cloud application creation. This helps application developers create cloud applications in a more productive way by letting them focus on cloud application development without needing to invest efforts into development environment setup and development tool management.

– PaaS provides a **service delivery platform**. A CSP provides service presence, orchestration, billing, mash-up and tools for associated development and testing by CSC through an application programming interface (API).

  NOTE 2 – Taking a CSC requesting a database on PaaS as an example, the service delivery platform receives the CSC's request, orchestrates and prepares the database service using proper resources on cloud computing, presents the cloud service to the CSC, ensures the cloud service's availability and reliability, maintains the relationship between CSC and the database it used, records the usage, generates bills, etc.

7 Overview of cloud native

7.1 Introduction to cloud native and cloud-native application

Cloud native is a methodology of software on cloud computing, which takes full advantage of capabilities of cloud computing during the process of software design, development, deployment,

running and management. This methodology is supported by principles and typical technologies, which are described in clause 7.2 and clause 7.3 in detail.

NOTE 1 – In ETSI [b-NFV-IFA 029], cloud native is defined as a software design principle with certain properties and a set of non-functional characteristics.

Being "cloud native" means an application grows on cloud computing and its lifecycle takes place within cloud computing as much as possible, following certain principles and using typical technologies. Cloud computing provides all about the application needs including cloud infrastructure capability, platform capability, security, maintenance, etc. except for application business logic code.

The goal of cloud native is to speed up application building, delivering, managing and optimization, making an application lightweight, flexible and automatic, as well as simplifying the work of the application developer and maintainer.

A cloud-native application [b-ISO/IEC TS 23167] is a type of cloud application that is explicitly designed to run within and to take advantage of the capabilities and environment of cloud services.

Figures 7-1 and 7-2 show the difference between traditionally developing an application and developing a cloud application following cloud native methodology.

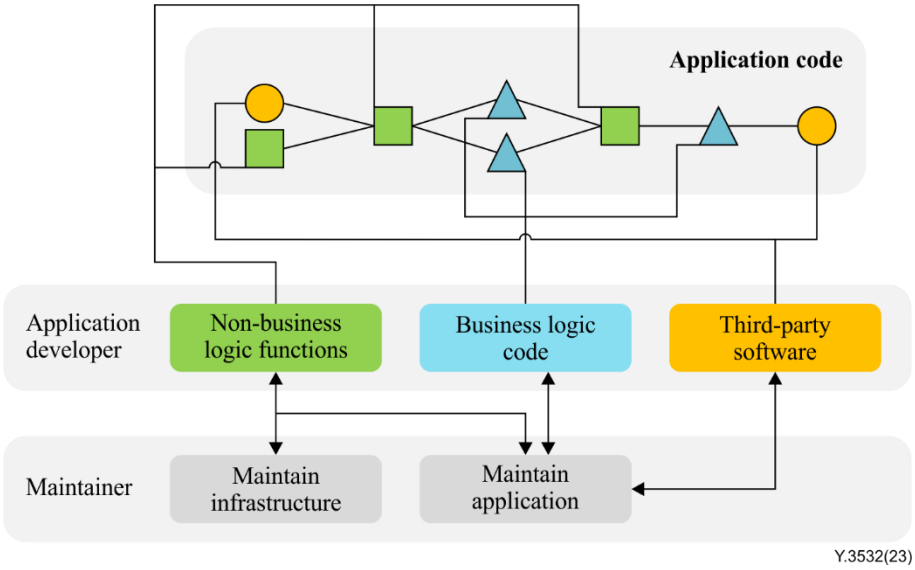


Figure 7-1 – Traditional way to develop and manage an application

Application code usually contains three parts: (1) business logic code, which implements core business logics; (2) non-business logic functions, which has have business logic but are necessary in a commercial product of customer-created applications, such as codes about alarms, logs and security, reliability; (3) third-party software, which is the reliance to implement application functions, for example, coding libraries, database software and firewall software.

As shown in Figure 7-1, in the traditional way of developing an application, application developers write all three parts of code. To manage a traditionally developed application, the application maintainer needs to take care of all stages and actions within the application's lifecycle, which include infrastructure management, application instance and status management, application lifecycle management etc. This creates difficulties for developers and maintainers in the development and management of applications. And as applications are growing and becoming more complicated, and application running environments vary, application developers and maintainers need to acquire more technologies, which increases the technical difficulty and human resource cost.

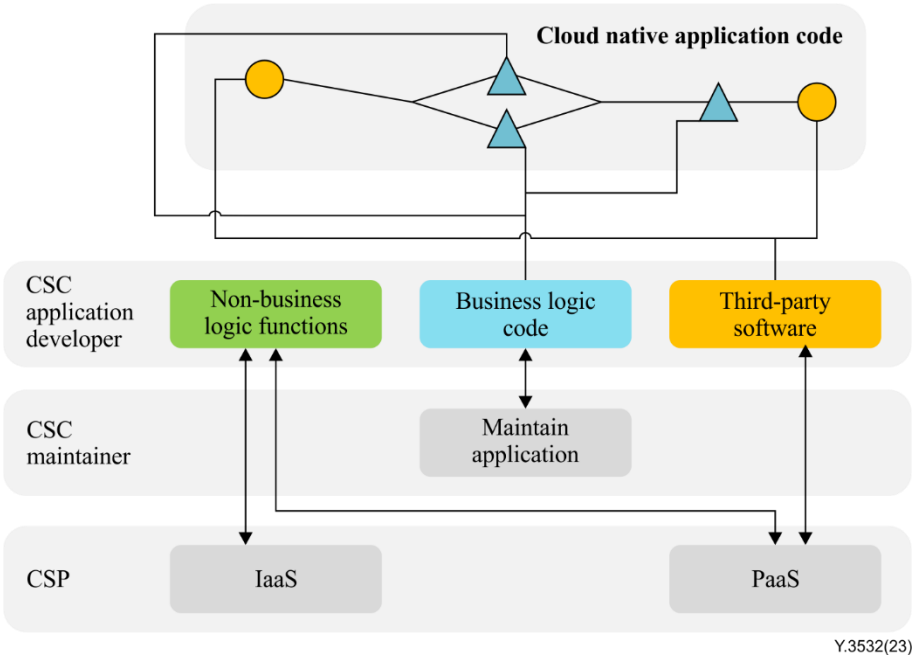


Figure 7-2 – Develop and manage a cloud application following cloud native methodology

In the cloud-native way of developing a cloud application, the CSC application developer only needs to write the business logic code and clarify how they want to use third-party software. Non-business logic functions and third-party software will be provided and managed by the CSP through IaaS and PaaS on cloud computing. This reduces the scope of technologies concerned by CSC application developers and leaves all complicated work beyond business logics to the CSP.

NOTE 2 – The complicated work beyond business logics dealt with by the CSP usually includes cloud service provision, reliability guarantee, scaling processing, security management, software upgrading, compatibility management, migration to multiple environments and process automation.

7.2 Principles to achieve cloud native applications

To better achieve cloud native, the following principles are usually followed when designing, developing, deploying, running and managing cloud native applications.

– **Use managed cloud services**: as most CSPs provide diverse managed cloud services, using these cloud services reduces the effort of application developer and maintainer to manage back-end software or infrastructure. To be cloud native, an application firstly uses cloud services.

  NOTE 1 – Managed cloud services are cloud services partially or completely managed by a CSP. Management objects include but are not limited to cloud service lifecycle, configuration, optimization, security and monitoring.

– **Design for automation**: as automation is a key feature of both the provision and the use of cloud services, cloud native suggests automatically testing, deploying, repairing and expanding applications. As an application grows and uses more cloud services, the complexity of application software integration, testing and delivery increases. With automation in an application lifecycle, more advantages of cloud native are revealed.

– **Handle the state of the application**: cloud native suggests designing stateless application components as much as possible by storing application state data (such as user data, business context data) in independent database. Being stateless makes it easy for application to scale, repair, rollback, load balance and migrate on cloud computing.

– **Be resilient**: Resilience is the ability of an application to provide defence against abnormal events and continue to run. Cloud native requires application resilience, which means the

application developer and maintainer consider the application's self-protection and redundancy design.

NOTE 2 – Examples of abnormal events include hardware failure, virtualization resource failure, resource exhaustion, software bug, hacking, insufficient business processing capability and power failure of the data centre.

NOTE 3 – Application self-protection includes monitoring status, performance and alerts of application, retrying the failed operation, limiting access to application components when suffer from heavy workload, break when failing and testing before delivery.

NOTE 4 – Redundancy includes deploying multiple replicas for important application components (e.g., using active/standby model), backing up database and deploying application in multiple availability zones/regions.

– **Zero trust**: applications in traditional architecture usually separate access entities into "trusted" entities and "un-trusted" entities, which are vulnerable to internal attacks and external threats. Cloud native practices the zero trust principle among all application components, which means every application component ensures its own security and resilience and gives zero trust to any entity outside itself. It helps applications, especially applications in distributed architecture, to be strong and easily deployed within cloud computing.

– **Be flexible in architecture**: as software requirements, technologies, organization structures and cloud services are always evolving, it is hard to define a perfect software architecture that would be permanently applicable. So being flexible in architecture make it easier to deal with change that may happen after an application is developed. This also helps applications to achieve faster development and delivery without waiting for complete functional design.

7.3 Technologies to achieve cloud native

The following typical technologies are preferred for providing cloud native applications:

– **Container**: As an isolated and portable execution environment for running software, container packages application software and all their dependencies, so that applications are no longer restricted by the environment and run in full functionality between different computing environments. A container allows cloud native application to easily create, scale and migrate on cloud computing.

NOTE 1 – As defined in [ITU-T Y.3535], a container is a set of software providing isolation, resource control and portability for virtualization processing of an application.

NOTE 2 – Choosing container as typical cloud native technology does not mean a container is the only choice. Both container and virtual machines on cloud computing are used to provide a cloud native application based on a CSC's requirements.

NOTE 3 – The reason of usage of container to providing a cloud native application is that a container has better portability than a virtual machine does.

– **Microservice**: A cloud native application using microservice is divided into multiple functional-independent modules that are rapidly deployed. This ensures the most flexible application software architecture. Any change that may be made to an application implemented as microservices is made to the lowest possible amount of application code by the smallest number of developers through the most flexible collaboration. And it also makes it possible for an application to use more cloud services.

– **Service mesh**: Service mesh provides mediation between all traffics among microservices for cloud native applications, and offloads microservice connections and interactions as a programmable infrastructure resource. Developers of cloud-native applications using service mesh no longer need to deal with network infrastructure but only focus on business logics. This supports making a cloud native application using microservices easier to use.

– **Automation tools**: Automation tools are a type of software providing continuous integration, continuous deployment and continuous delivery capabilities to cloud native applications, which form automation pipelines and reduce the management difficulty of lifecycle for cloud native application in flexible architecture.

7.4 Relationship between a cloud native application and PaaS

To use as many cloud services as possible, as the cloud native application desires, major phases of cloud native application lifecycle (including coding, testing, deployment and maintenance) happen preferably within cloud computing, which is perfectly satisfied by PaaS.

For example, PaaS's IDE and development tools allow the CSC to carry out coding and testing cloud native applications on cloud computing. PaaS's application hosting environment allows the CSC to obtain everything (besides application business logic codes) for cloud native application deployment and running, including physical and virtual resources and application dependent software. The PaaS's service delivery platform takes care of PaaS service management and relieves the CSC of complicated cloud service operations. PaaS is the cloud service chosen most often by cloud native applications.

Taking on-premise application as a comparison, a CSC needs to provide and manage everything needed within the application lifecycle, which includes the application development environment, development tools, application hosting environment, etc. As the CSC uses no cloud service, on-premise application is not a cloud native application.

Taking a cloud application using an IaaS service as a comparison, the CSC uses only infrastructure resources provided by the CSP, while the CSC still needs to provide and a manage partial application hosting environment (operating system, runtime, etc.), application development environment, development tools, etc. Using IaaS is necessary but insufficient to implement a cloud native application.

8 Functional requirements of PaaS for cloud native applications

8.1 Requirements of PaaS for cloud native application deployment

– **(Application execution environments)** It is required that the PaaS CSP provides application execution environments to the CSC.

NOTE 1 – Application execution environment includes physical and virtualized resources (bare metal, virtual machine and container), operating systems, runtimes, middleware, application dependent software, etc.

– **(Application template)** It is recommended that the PaaS CSP provides an application template for the CSC to describe cloud native application for deployment.

NOTE 2 – Application template is a file with a standard format to describe microservice architectures, cloud native application images, required resources, configurations, etc.

NOTE 3 – An application template is used for application deployment.

– **(Application image repository)** Image repository It is recommended that the PaaS CSP provides cloud application image repository to support uploading, storing, updating and deleting images of cloud native application by the CSC.

– **(Application template repository)** It is recommended that the PaaS CSP provides application template repository to support uploading, storing, updating and deleting application templates by the CSC.

– **(Application template conversion)** It is recommended that PaaS CSP provides a conversion on application templates, which would be at least converted into resource requirements and configurations on resources and application.

8.2 Requirements of PaaS for cloud native application functional support

– **(Microservice registry)** It is recommended the PaaS CSP provides a microservice registry to the CSC to support the management of microservices for cloud native applications.

NOTE 1 – A microservice registry is a functional module that maintains a map of microservice name and its real-time IP address, which lets microservices access each other using only the microservice name. Through the registration of the microservice to the registry, the CSC and the CSC's application discover registered microservices from the registry.

– **(Load balancing proxy)** It is recommended that the PaaS CSP provides a load balancing proxy to the CSC to expose a unified access point of the microservice-based cloud native application and distribute access traffic to microservices.

NOTE 2 – A load balancing proxy is an entity to publish an entry point of a cloud native application to support load balancing of traffic.

– **(Configuration of load balancing proxy)** It is recommended that the PaaS CSP provides load balancing rule configuration of load balancing proxy to the CSC.

NOTE 3 – An example of load balancing configuration is a traffic distribution policy.

– **(Traffic controller)** It is recommended that the PaaS CSP provides a traffic controller to the CSC to control traffic among microservices of the cloud native application.

NOTE 4 – A traffic controller is provided by a proxy, load balancer, service mesh, etc. A traffic controller is recommended to support a traffic management rule including rate limiting, circuit breaking and timeout.

NOTE 5 – To manage interactions among microservices, a traffic controller is required to monitor connections of microservices.

– **(Traffic control configuration)** It is recommended that a PaaS CSP provide configuration of the traffic controller to the CSC.

NOTE 6 – Configuration features of a traffic controller include rate limitation, biggest number of requests before circuit break, length of waiting time of a request, etc.

– **(Application data storage)** It is recommended that the PaaS CSP provide data storage for cloud native applications to the CSC.

NOTE 7 – Examples of data storage are a hierarchical database, relational database and non-relational database.

– **(Application monitoring)** It is recommended that the PaaS CSP provide monitoring of the cloud native application to the CSC.

NOTE 8 – The monitoring outcome of the cloud native application is metrics.

NOTE 9 – Metrics are a group of performance data to indicate the running status and quality of a system. For example, for a web application, the monitored metrics are the quantity of processed website traffic, visitor number, etc. The metrics for monitoring a cloud native application are defined by application developer and generated by application at running status.

– **(Alerting application status)** It is recommended that the PaaS CSP provide alert of the application status to the CSC based on the monitoring of the cloud native application.

NOTE 10 – To generate alarms, it is needed to set a threshold of application monitoring metrics by theCSC.

– **(Application tracing)** It is recommended that PaaS CSP provide tracing of traffic flows within a cloud native application to the CSC.

NOTE 11 – Tracing is used by the developer and operator of a cloud native application to detect the failure point when the cloud native application is not working properly. Tracing includes tracking the complete route of target traffic flow, collecting vital information including flow sender, flow receiver, time consumption of a microservice processing the flow, etc.

– **(Tracing feature configuration)** It is recommended that the PaaS CSP provide a tracing feature configuration to the CSC.

NOTE 12 – A tracing feature includes target tracing IP address, target tracing protocol, tracing time duration, target microservice, etc.

– **(Application logging)** It is recommended that the PaaS CSP provide log management on the cloud native application to the CSC.

NOTE 13 – The log is generated by the cloud native applications. Log management includes log collection, storage, deletion and export of logs. Examples of logs are the application system logs and operations logs, which record the activities conducted by the cloud native application.

8.3 Requirements of PaaS for cloud native application development

– **(Application reliability testing)** It is recommended that the PaaS CSP provide a reliability test of a cloud native application to the CSC to detect potential but non-enumerable failure.

NOTE 1 – The example of application reliability testing is to check the robustness and reliability of application system by manually creating unstable features in the application system and checking the reaction of application system. The unstable features vary from application to application; examples include restricting access, forcing failover, forcing system clocks out of synchronization, significantly increasing application workload, etc.

– **(IDE)** It is recommended that the PaaS CSP provide multiple-language IDEs to support the CSC in developing cloud applications on PaaS.

NOTE 2 – Multiple-language IDEs are developing environments for different coding languages.

NOTE 3 – An IDE usually includes a code editor, code compiler, debugger and graphic user interface.

– **(Software development kits)** It is recommended that the PaaS CSP provide software development kits to the CSC to support cloud native application coding.

NOTE 4 – A software development kit contains software framework, libraries, functions and tools.

– **(Testing tool)** It is recommended that the PaaS CSP provide testing tools for the cloud native application to the CSC.

NOTE 5 – Common testing tools convert the unit test, interface test, functional test, performance test, etc.

– **(Continuous deployment and testing)** It is recommended that the PaaS CSP provide a continuous deployment and testing pipeline to the CSC to achieve fast cloud native application delivery.

NOTE 6 – A continuous deployment and testing pipeline is a system to automatically deploy and test an application every time the CSC updates the application code.

– **(Development progress management)** It is recommended that the PaaS CSP provide development progress management of the cloud native application to the CSC.

NOTE 7 – Development progress management tracks the application name, development stage and information of the team responsible for development tasks.

NOTE 8 – Development stages include requirement analysis, planning, designing, developing, testing and releasing the cloud native application.

– **(Bug tracking)** It is recommended that the PaaS CSP provide bug tracking to the CSC to track software bugs of the cloud native application and the software bug solving status.

NOTE 9 – A software bug is an error, flaw or fault in the creation of software. A software bug causes computer software to produce unexpected results or behave in unintended ways.

– **(Code management)** It is recommended that the PaaS CSP provide code management to the CSC for code storage, code version management, code developer information management and code update record management.

8.4 Management requirements of PaaS

– **(Service catalogue)** It is required that the PaaS CSP provide a service catalogue to deliver a list of PaaS for CSC subscription.

NOTE 1 – A service catalogue includes detailed service information such as service name, service description, service image location and service user manual.

– **(Unified service release procedure)** It is required that the PaaS CSP provide a service release procedure to publish a newly developed PaaS service.

NOTE 2 – A service release procedure of a PaaS is a set of processes to publish a CSP-developed software. After the service release procedure, this new PaaS service is available in the service catalogue as well as being subscribed to by the CSC.

NOTE 3 – A service release procedure usually includes service images and service deployment file uploading and storage, service security verification, service availability checking, service introduction information filling and adding the service-to-service catalogue.

– **(Service subscription list)** It is required that the PaaS CSP provide subscription information, which at a minimum includes information on the CSC, the PaaS subscribed to by the CSC and the subscription time.

– **(Service instantiation)** It is required that the PaaS CSP provide service instantiation of the PaaS after being subscribed to by the CSC.

NOTE 4 – The service instantiation of the PaaS includes selecting corresponding images and deployment files of the target service, selecting target type and number of resources, service deployment and service configuration.

– **(Service access control)** It is required that the PaaS CSP provide control of CSC access to the PaaS.

– **(Service upgrading management)** It is required that the PaaS CSP provide service upgrading management on the PaaS.

NOTE 5 – Service upgrading management of PaaS includes PaaS service version control, PaaS service upgrading progress tracking, rollback control when failing to upgrade, load balancing control on traffics between an old-version PaaS service and new-version PaaS service, CSC notification about PaaS service upgrading progress, etc.

– **(Service instance deleting)** It is required that the PaaS CSP provide deletion of the PaaS service instance.

NOTE 6 – By deleting a PaaS service instance, the CSP stops the access of the CSC to the PaaS service, deletes the service instance and relevant data from the database and releases physical and virtual resources used by the service instance.

– **(Service with multiple resource types)** It is recommended that the PaaS CSP provide the selection of a PaaS service according to multiple resource types.

NOTE 7 – Multiple resource types include container, virtual machine and bare metal. A PaaS service is instantiated with any of the resource type.

– **(Service metrics monitoring)** It is required that the PaaS CSP provide monitoring of metrics of the PaaS service instance to reflect service status and performance.

NOTE 8 – The metrics of the PaaS service instance are used by both the CSP and CSC to know whether a PaaS service works properly. It usually includes service-level metrics and resource-level metrics.

NOTE 9 – Resource-level metrics are metrics of physical and virtual resources used by the PaaS service instance. Typical resource-level metrics include the CPU rate, memory rate, disk rate, network throughput, threads per CPU, etc.

NOTE 10 – Service-level metrics are usually defined by the PaaS service developer, and vary from service to service. For a database, metrics include database throughput, database response time, number of query errors, etc.

– **(Service metrics management)** It is required that the PaaS CSP provide management of the metrics of PaaS service instance subscribed to by the CSC.

NOTE 11 – The management of metrics of the PaaS service instance includes storage, deletion, searching and exporting metrics of the PaaS service instance.

– **(Service metrics storage duration management)** It is recommended that the PaaS CSP provides setting the time length of metrics of the PaaS service instance, beyond which metrics would be automatically deleted.

– **(Alerts on service metrics)** It is recommended that the PaaS CSP provide alerting on unexpected metrics for the PaaS service instance.

– **(Service metrics report)** It is recommended that the PaaS CSP provide reports on metrics monitoring the PaaS service instance.

NOTE 12 – The monitored result of the PaaS service instance is reported periodically or automatically.

– **(Metrics dashboard)** It is required that the PaaS CSP provide a dashboard to display metrics of PaaS service instance.

– **(Service log management)** It is required that the PaaS CSP provide log management of the PaaS service instance.

NOTE 13 – Log management includes collecting, storing, deleting, indexing and exporting logs of the PaaS service instance and corresponding resources automatically.

– **(Log storage duration management)** It is recommended that the PaaS CSP provide setting on the time length of logs of the PaaS service instance, beyond which logs will be automatically deleted.

– **(Service image management)** It is required that the PaaS CSP provide management of PaaS images, which includes storage, updating, deleting, searching and multiple version management.

– **(Service deployment files management)** It is recommended that the PaaS CSP provide management of deployment files of the PaaS service, which include storage, updating, deleting, searching and multiple version management.

NOTE 14 – A deployment file is a set of codes to instruct a computer how to deploy a software.

– **(Pull remote image and deployment files)** It is recommended that PaaS CSP provide pulling of PaaS service images and deployment files from remote repositories and storing a copy for future usage.

9 Security considerations

Security aspects for consideration within cloud computing environments including PaaS are addressed by security challenges for CSPs, as described in [ITU-T X.1601]. In particular, [ITU-T X.1601] analyses security threats and challenges and describes security capabilities that could mitigate these threats and meet security challenges.

Appendix I

Use case of cloud native PaaS

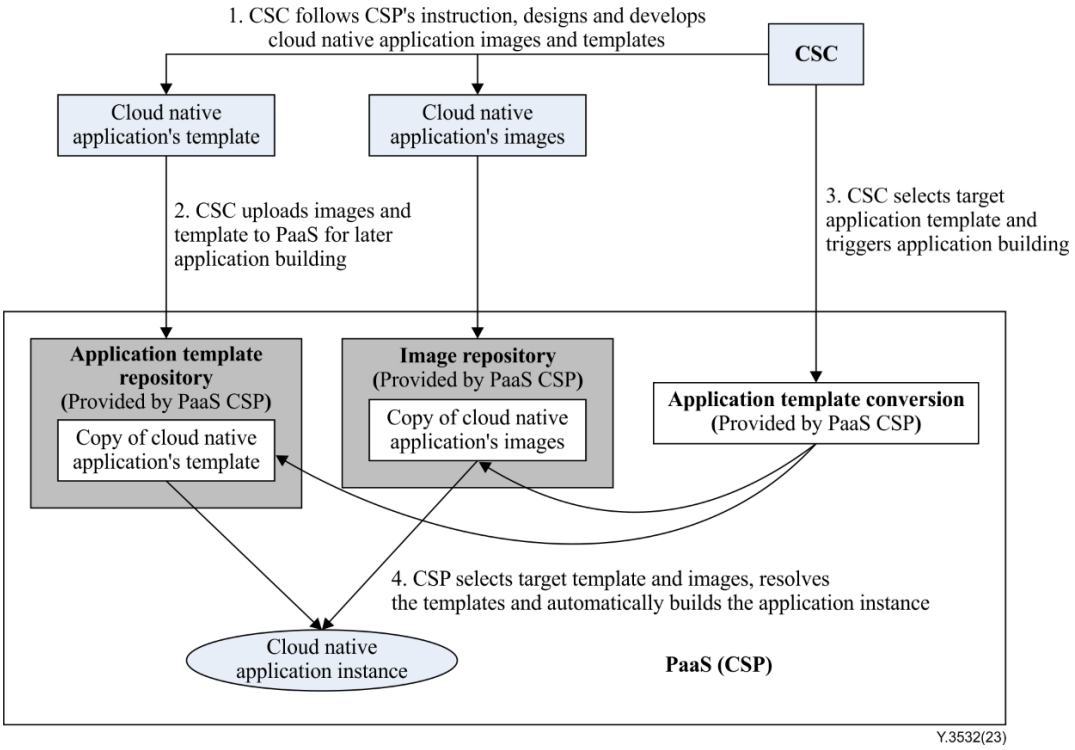
(This appendix does not form an integral part of this Recommendation.)

I.1 Use case: cloud native application deployment support

Table I.1 – Use case: cloud native application deployment support

Title	Cloud native application deployment support
Description	This use case describes the PaaS CSP support of the CSC in cloud native application deployment. As cloud native applications are usually designed and deployed on the cloud, it is necessary to use a platform capability to simplify the building and deploying of applications. This is different from building applications using infrastructures such as virtual machines or bare metals, which requires developers to manually deploy applications or use a pre-developed deployment script. With PaaS, the application deployment process is automatic and simple. Firstly, the PaaS would provide the CSC a standard application template through which the CSC describes the application to the PaaS, giving the application's microservices relationship, resource requirements, configurations and required application images, etc. The PaaS platform also provides an application template repository to help the CSC manage application templates, including uploading, storing, deleting, updating, etc. As applications are usually deployed from images, the PaaS provides an image repository to let the CSC upload, store, update and delete customized application images. Then, after receiving the CSC's deployment command, the PaaS platform would automatically resolve the template, obtain resource requirements, and automatically instantiate the application once receiving the CSC's command. This would derive a requirement for application template conversion.
Roles	CSP, CSC

Table I.1 – Use case: cloud native application deployment support

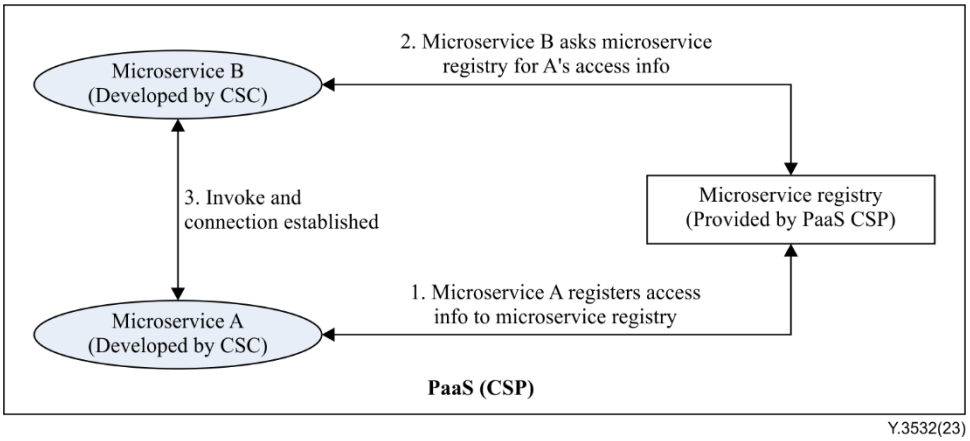
Title	Cloud native application deployment support
Figure (optional)	 **Figure I.1 – Using PaaS for cloud native application instantiation**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application template (refer to clause 8.1) – Application image repository (refer to clause 8.1) – Application template repository (refer to clause 8.1) – Application template conversion (refer to clause 8.1) – Application execution environments (refer to clause 8.1)

I.2 Use case: microservice support

Table I.2 – Use case: microservice support

Title	Microservice support
Description	Cloud native applications are usually designed and developed as a group of microservices. A cloud native application could be separated into several software modules, and each of the modules is developed as a single microservice. The number of instances running in the environment would increase significantly, which makes it impossible for each microservice to know the access address of the target microservice. Usually, the access addresses of microservices are not hardcoded with application code,

Table I.2 – Use case: microservice support

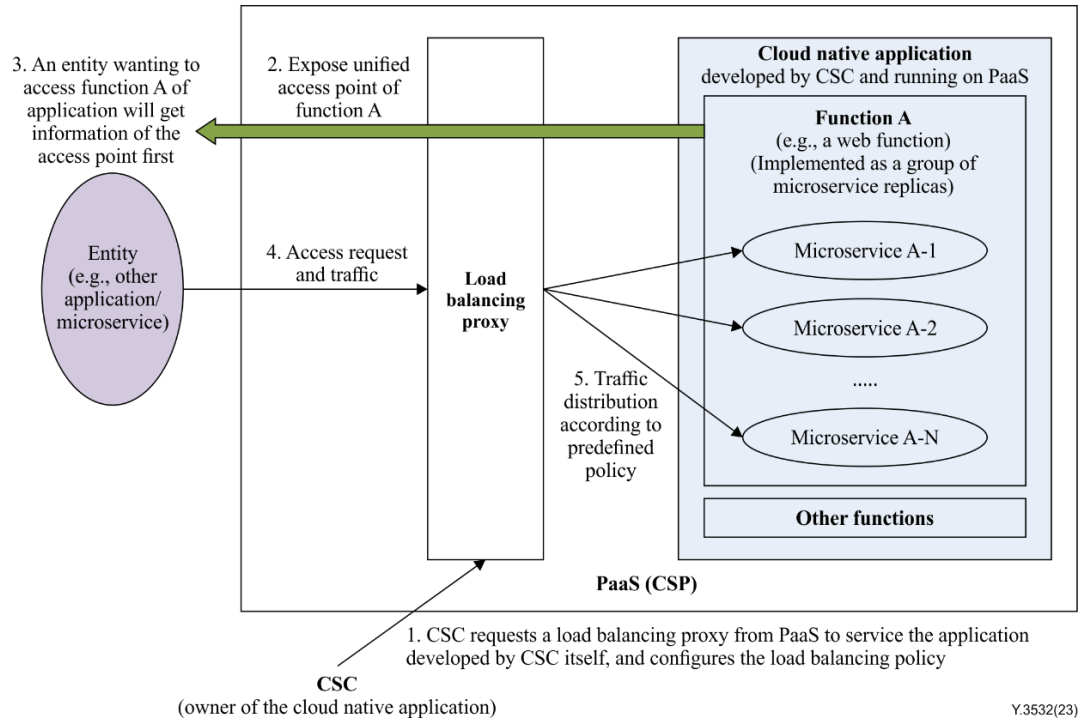
Title	Microservice support
	which makes it better to obtain a microservice address with a public address management centre. A PaaS for cloud native applications needs to run these microservices and manage their access addresses and establish connections among them, of which the detailed communication process is described as follow: – A CSC developed an application with two microservices (A and B) and deployed them on the execution environment of the PaaS. – Microservice A and microservice B would firstly register themselves in microservice registry provided by PaaS CSP to expose themselves within the execution environment of PaaS. – Microservice B needs to communicate with microservice A. – In order to find microservice A, microservice B would go to the microservice registry to discover (usually asking for the internal IP address) microservice A. – Then microservice B invokes A and communicate with A. The microservice registry would also monitor the running status of microservice A. If it is not available, the microservice registry will respond to microservice B with a negative connection.
Roles	CSP, CSC
Figure (optional)	 **Figure I.2 – Using PaaS for microservice of cloud native application**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Microservice registry (refer to clause 8.2)

I.3 Use case: load balancing

Table I.3 – Use case: load balancing

Title	Load balancing
Description	As cloud native applications are usually designed and developed as a group of microservices, multiple microservice replicas to achieve the same function would exist,

Table I.3 – Use case: load balancing

Title	Load balancing
	which expands processing capability. So, cloud native application usually needs a load balancer for microservice access. PaaS for cloud native applications usually provides a load balancing proxy to the CSC to achieve microservice load balancing. The load balancing proxy will provide a unified access point to a group of microservice replicas with the same function, and distribute traffic to replicas following a pre-defined policy. The CSC needs to configure a load balancing policy of the load balancing proxy provided by the CSP to ensure that traffic is distributed based on the CSC's requirements.
Roles	CSP, CSC
Figure (optional)	 **Figure I.3 – Using PaaS for load balancing for microservice replicas of cloud native application**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Load balancing proxy (refer to clause 8.2) – Configuration of load balancing proxy (refer to clause 8.2)

I.4 Use case: traffic controller

Table I.4 – Use case: traffic controller

Title	Traffic controller
Description	As cloud native applications are usually designed and developed as a group of microservices, huge amounts of microservice instances could exist. The traffic interactions among microservices within this environment are too complicated to be handled either by microservices themselves or by CSC developers. It is inconvenient to code about every possible traffic fault and traffic control policy. A traffic controller provided by a PaaS helps developers to easily manage traffics among microservices. When a fault occurs, a traffic controller helps microservices to maintain resiliency by properly controlling traffic. Different types of control might be needed by microservices through a traffic controller: – Rate limiting: The processing capability of an example microservice, A, is limited. If the number of requests to microservice A exceeds its processing limit, microservice A would be overwhelmed. The CSC uses a traffic controller to play as a proxy for microservice A and to set a limitation for traffic sent to microservice A. For those requests exceeding the processing limit, the traffic controller declines those requests or holds them for later processing. This helps microservice A maintain a relatively stable running and processing status (Figure I.4-1). – Circuit breaking: During the interactions of microservices, faults may occur. Microservice A may respond slowly or be temporarily unavailable. In this condition, if microservice B requests microservice A, the CSC uses the traffic controller to play a proxy for microservice A and set a circuit breaking policy to simply reject the request or return an exception. This helps to prevent microservice B from continuous requesting. When microservice A works appropriately, a traffic controller will play a transparent proxy between microservice A and B (Figure I.4-2). – Timeout: Microservice B sends a request to microservice A and waits for a respond in a timely manner. If microservice A does not respond in a long time, CSC uses traffic controller and set timeout policy to send exceptions to prevent microservice B from waiting (Figure I.4-3). As the traffic controller manages the traffic, it is recommended to monitor the connections and traffic among microservices. As traffic control logics are designed by application developers, which are CSCs, a CSC needs to configure the traffic management policies of the traffic controller so that the CSC controls interactions among microservices. The policies include but are not limited to: rate limiting, circuit breaking, timeout.
Roles	CSP, CSC

Table I.4 – Use case: traffic controller

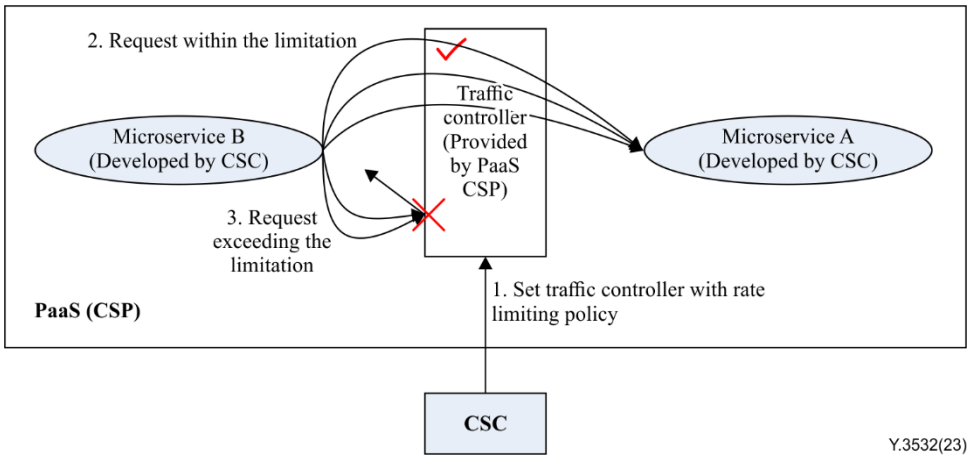
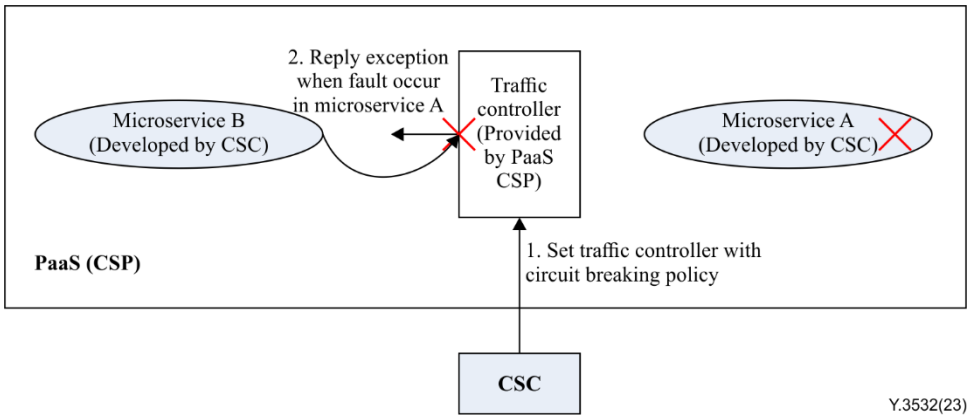
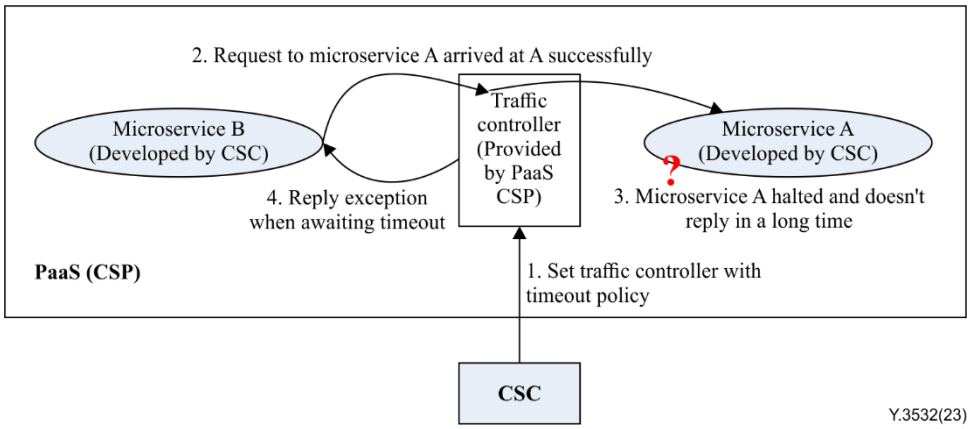
Title	Traffic controller
Figure (optional)	 **Figure I.4-1 – PaaS providing traffic controller for rate limiting**  **Figure I.4-2 – PaaS providing traffic controller for circuit breaking**  **Figure I.4-3 – PaaS providing traffic controller for timeout**
Pre-conditions (optional)	
Post-conditions (optional)	

Table I.4 – Use case: traffic controller

Title	Traffic controller
Derived requirements	– Traffic controller (refer to clause 8.2) – Traffic control configuration (refer to clause 8.2)

I.5 Use case: application data storage

Table I.5 – Use case: application data storage

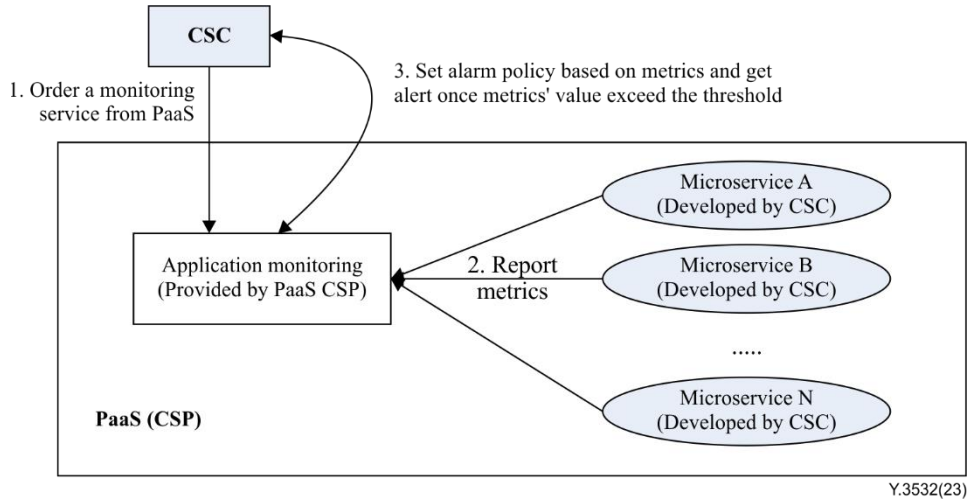
Title	Application data storage
Description	Within an application, there exists large amount of data. For example, if the application is an email system, this application would as a minimum have the data of users' email address and password. If the application is a 5G network function, then this application would as a minimum have users' registration data, users' SLA data, users' phone number and users' identity data. As cloud native applications are preferably developed on PaaS using PaaS services to the greatest extent possible, the data storage service provided by PaaS to CSC supports CSC storage application data.
Roles	CSC, CSP
Figure (optional)	
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application data storage (refer to clause 8.2)

I.6 Use case: application monitoring

Table I.6 – Use case: application monitoring

Title	Application monitoring
Description	As a CSC needs to know the health status of their cloud native application systems, monitoring is one of the most important functions that all applications would have. There are many mature open-source monitoring software such as Prometheus which could be provided by a PaaS. So, it is common for cloud native applications to apply for a monitoring service on a PaaS. With a monitoring service, the CSC monitors the metrics of cloud native applications and obtains a metrics visualization view including the number of requests served, tasks completed, errors, the number of currently running microservices, histogram, response duration, etc. CSC uses this monitoring data and generates an alarm based on metrics, so that the CSC is alerted when metrics are not correct in application systems.
Roles	CSP, CSC

Table I.6 – Use case: application monitoring

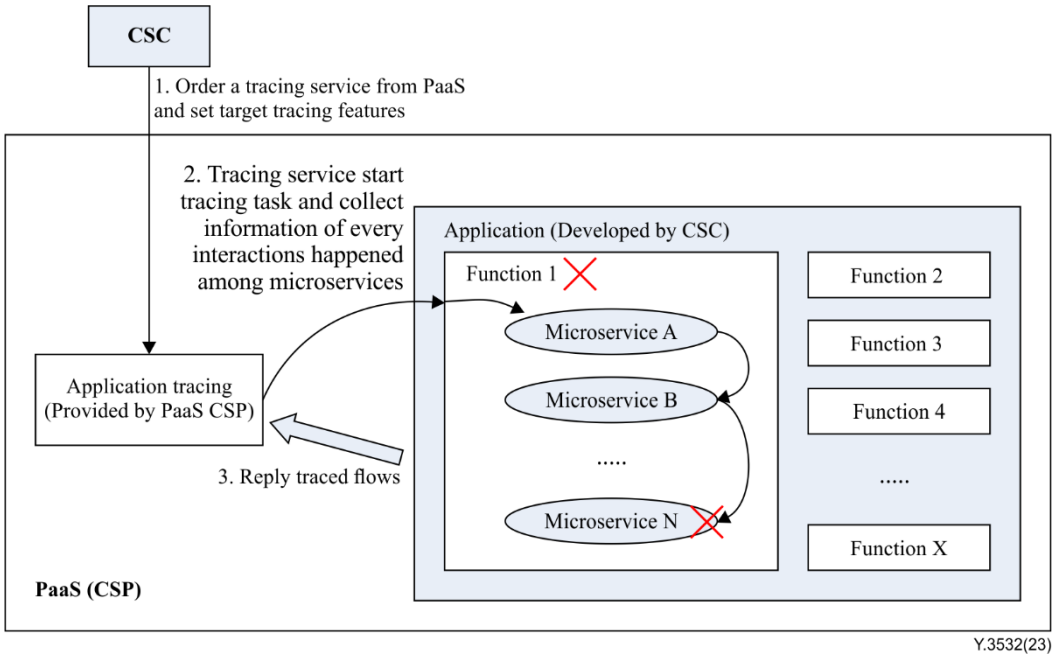
Title	Application monitoring
Figure (optional)	 **Figure I.6 – Utilization of application monitoring of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application monitoring (refer to clause 8.2) – Alerting application status (refer to clause 8.2)

I.7 Use case: application tracing

Table I.7 – Use case: application tracing

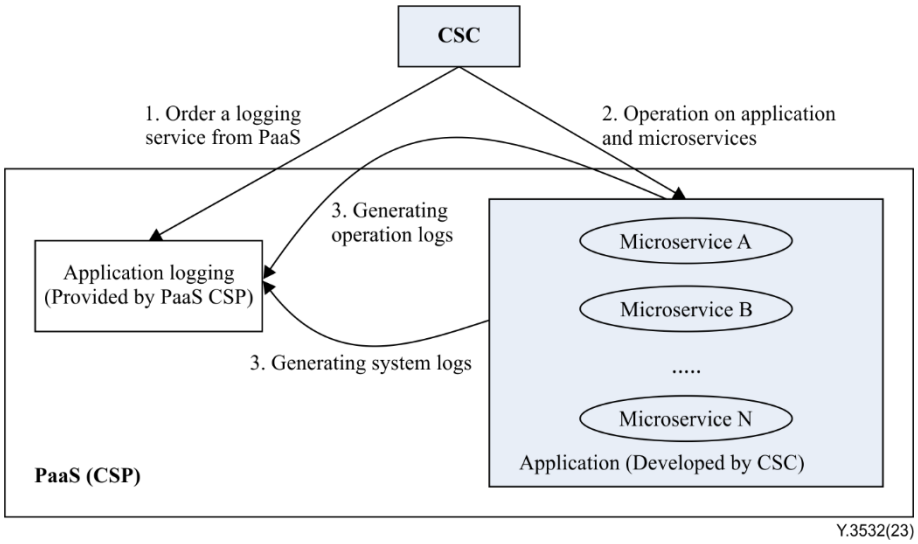
Title	Application tracing
Description	Cloud native applications usually have several functions and each function would consist of several microservices. There exists a huge amount of internal and external interactions among those microservices. A function of an application may not work correctly. To find out which step causes the fault, a CSC needs a traffic tracing service to track the interactions taking place among all related microservices. With the traffic tracing service, the CSC obtains the information of received/sending request, received/sending response, time consumption of request and response, etc., which helps the CSC analyse which step is wrong. The application tracing capability provided by the CSP helps the CSC to collect microservice relationship and track traffic flows happening in applications. Once a fault occurs, this service helps the CSC to make the analysis. To increase the tracing accuracy, the CSP supports tracing feature configuration so that the CSC receives highly related tracing information.
Roles	CSP, CSC

Table I.7 – Use case: application tracing

Title	Application tracing
Figure (optional)	 **Figure I.7 – Utilization of application tracing of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application tracing (refer to clause 8.2) – Tracing feature configuration (refer to clause 8.2)

I.8 Use case: application logging

Table I.8 – Use case: application logging

Title	Application logging
Description	For the CSC's cloud native applications, there are many actions, operations, events happening within the application system. The CSC needs to know when the microservices are to be deployed, debugged, started, stopped, failed and redeployed, as well as who made the operation. And during the running process, the application itself is also designed to generate logs to reflect its running status. The above information is collected as logs. The cloud native applications would generate logs. An application logging capability provided by PaaS CSP helps to store and manage those logs. The PaaS CSP also provides the visualization of cloud native application logging and relevant resource logging for CSC review and debug.
Roles	CSP, CSC
Figure (optional)	 **Figure I.8 – Utilization of application logging of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application logging (refer to clause 8.2)

I.9 Use case: weakness detection of application

Table I.9 – Use case: weakness detection of application

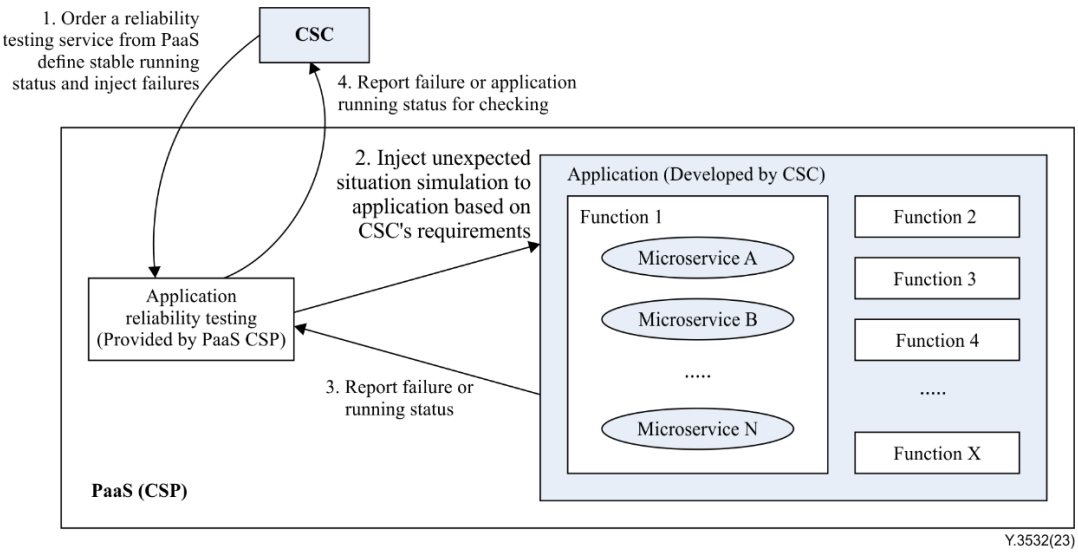
Title	Weakness detection of application
Description	The CSC needs to predict unexpected system failure to attain consistent reliability. A cloud native application usually has a complicated structure (which may have tens of thousands of microservices) and runs in a complicated environment. The error or failure of the application may be for many unpredictable reasons, for example resource failure, application microservice failure, PaaS service failure and burst traffic. It is hard to use a unit test or integrating test or functional test or some other traditional testing methods to test all possible scenarios that could cause application failure. The failure can happen at any time under any circumstance. One of PaaS's objectives is to help the CSC to build reliable applications, which means it provides a tool helping the CSC to test application reliability under any expected and unexpected failures. The CSP provides application reliability testing by deliberately injecting faults that cause system failures, for example taking dependencies offline (stopping API apps, shutting down VMs, etc.), restricting access (setting firewall rules), forcing failover, forcing system clocks out of synchronization with each other, etc.
Roles	CSP, CSC
Figure (optional)	
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Application reliability testing (refer to clause 8.3)

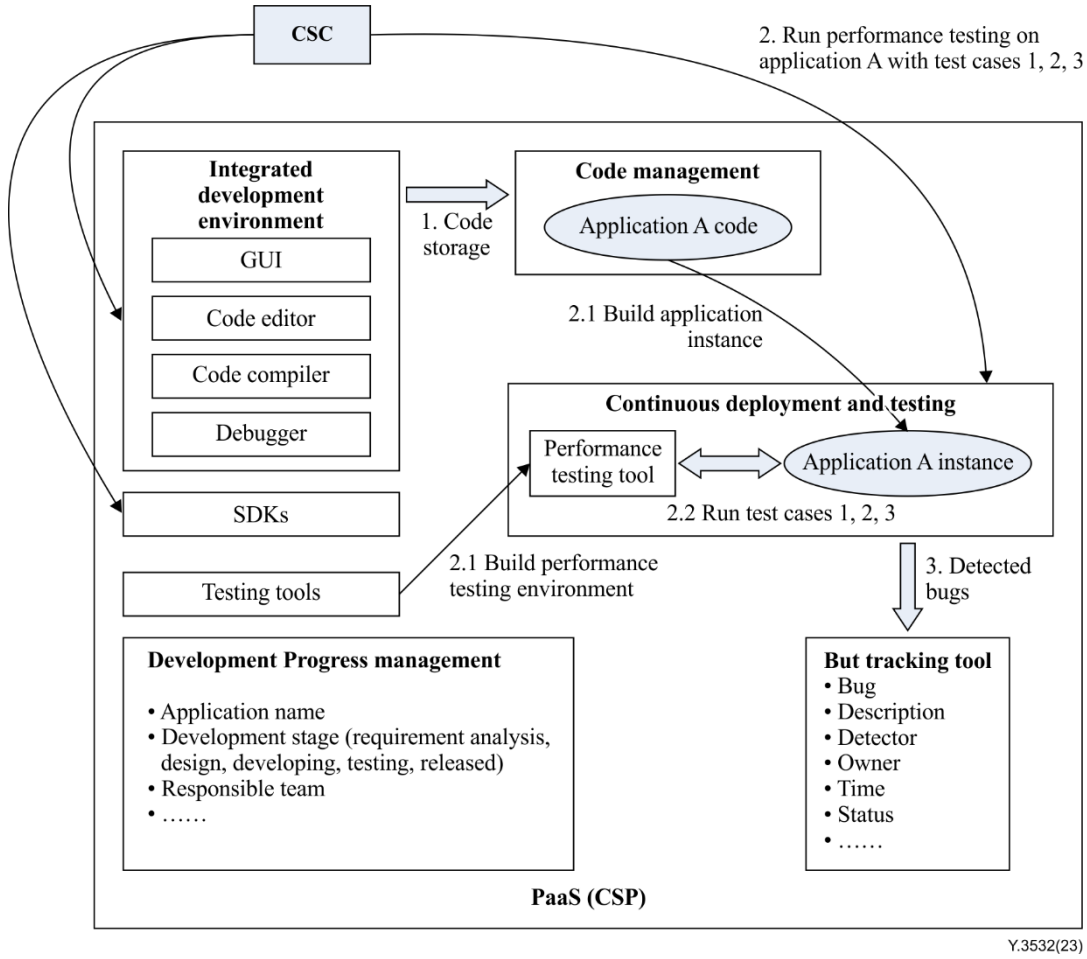
Figure I.9 – Utilization of application reliability testing of PaaS

I.10 Use case: application development support

Table I.10 – Use case: application development support

Title	Application development support
Description	As cloud native applications are to be directly developed and tested on cloud computing where possible, a PaaS CSP assists the CSC with development. Usually, when a CSC develops an application, it requires an IDE which includes a code editor, code compiler, debugger and graphic user interface. Then a software development kit is also needed by the CSC to provide the required software framework, libraries and functions. Besides an integrated development environment, all those development tools integrated together are separately provided by the CSP to the CSC. After developing the code, testing is another important process for the CSC to check the codes and application. Within a testing period, CSC may require testing tools to cover a unit test, interface test, functional test and performance test. As many tests would be necessary, a CSC needs to deploy the developed application many times and run corresponding tests many times in the development and testing environment. To simplify the overall deployment and test running process, it is preferred to have a continuous deployment and testing pipeline. This pipeline automatically deploys the application and runs test cases every time the CSC requires. To better manage the development progress, the CSC also needs progress management tools to track development progress, bug management tools to track all the bugs detected in the application and their status in terms of their solution. As software usually continues to be upgraded, multiple versions of one application would exist. A CSC will need code version management tools.
Roles	CSC, CSP

Table I.10 – Use case: application development support

Title	Application development support
Figure (optional)	 **Figure I.10 – PaaS support development of cloud native applications**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Integrated development environment (refer to clause 8.3) – Software development kits (refer to clause 8.3) – Testing tool (refer to clause 8.3) – Continuous deployment and testing (refer to clause 8.3) – Development progress management (refer to clause 8.3) – Bug tracking (refer to clause 8.3) – Code management (refer to clause 8.3)

I.11 Use case: service catalogue management of PaaS

Table I.11 – Use case: service catalogue management of PaaS

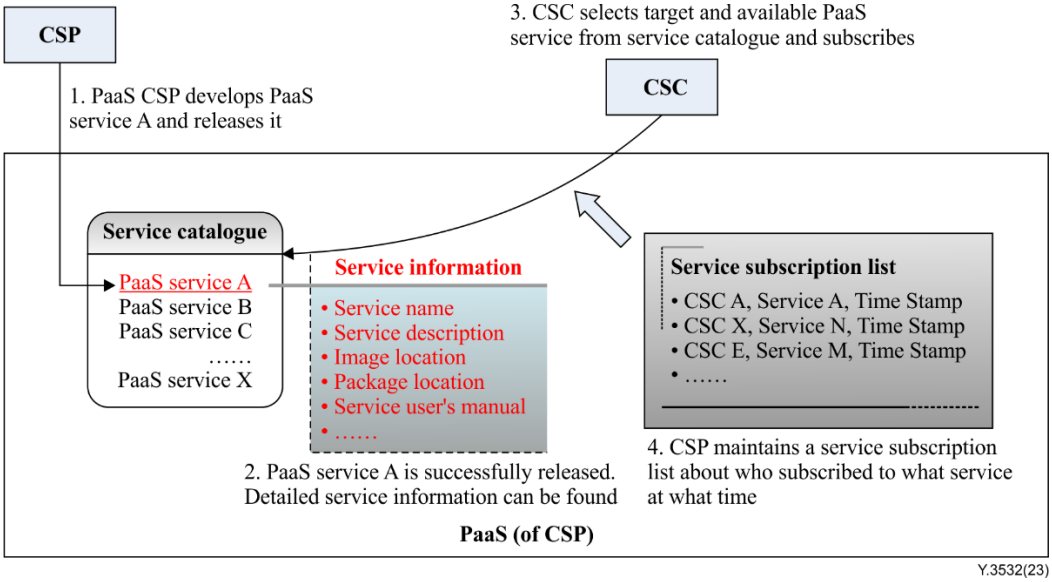
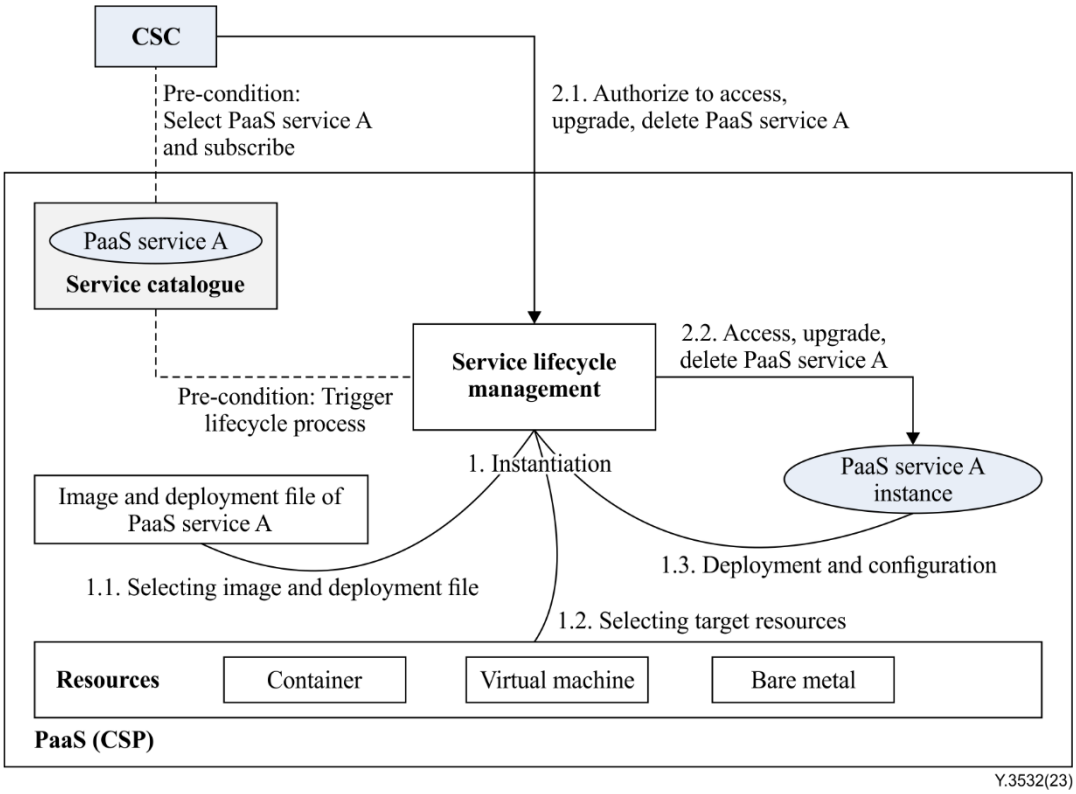
Title	Service catalogue management of PaaS
Description	This use case describes how the PaaS CSP provides management and maintenance on a catalogue of all PaaS services. A PaaS usually provides diverse services such as load balancing proxy, traffic controller, monitoring on application and logging on application. This requires that the CSP provide a service catalogue which maintains the PaaS service list and service information, controls service availability and manages service subscription. Figure I.11 explains the details. – Step 1: Usually PaaS CSP would take care of cloud service development and service release. PaaS CSP follows a unified service release procedure to make the developed PaaS service available to the CSC. The procedure includes service image and deployment file uploading, service information filling, service security verification, etc. – Step 2: If the new PaaS service is successfully released, within the service catalogue, this new PaaS service will be added and detailed service information will be displayed. – Step 3: CSC subscribes target a PaaS service from the service catalogue of the PaaS. After the subscription is established, the CSP maintains a service subscription list to track who subscribed to which service at what time.
Roles	CSP, CSC
Figure (optional)	
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Service catalogue (refer to clause 8.4) – Unified service release procedure (refer to clause 8.4) – Service subscription list (refer to clause 8.4)

Figure I.11 – Utilization of service catalogue of PaaS

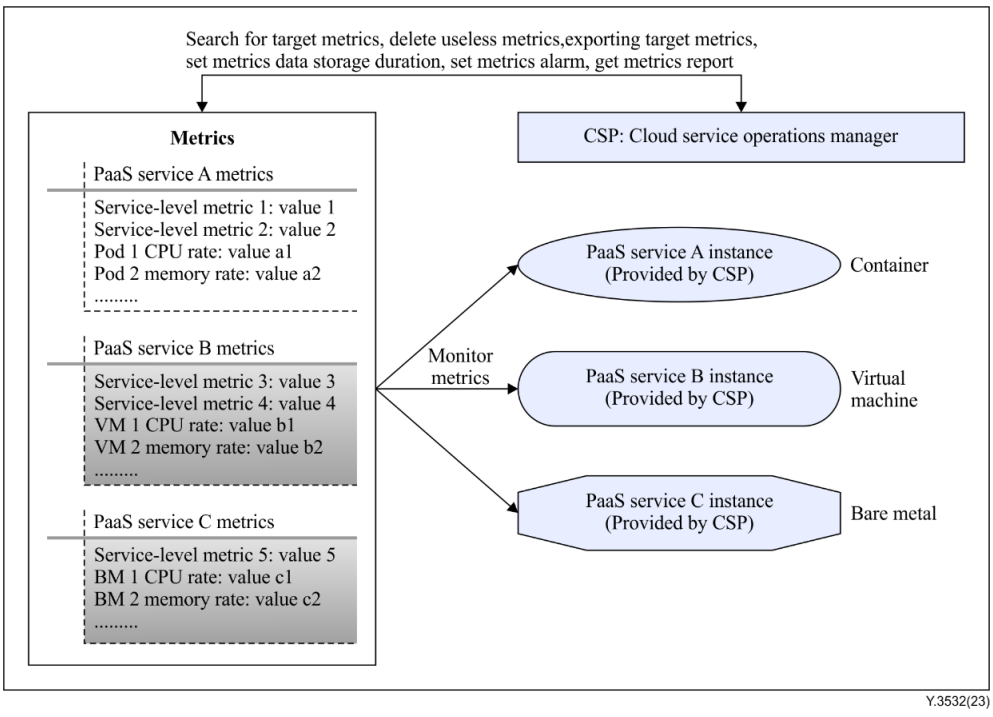
I.12 Use case: service lifecycle management of PaaS

Table I.12 – Use case: service lifecycle management of PaaS

Title	Service lifecycle management of PaaS
Description	This use case describes how the CSP manages the service lifecycle of PaaS. To provide a service instance to the CSC based on a subscription, the CSP is able to manage the service lifecycle, which includes service instantiation, service access authorization, service upgrading and service deleting. As cloud native application usually uses containers as a resource, the PaaS services required by CSCs for their cloud native applications are usually preferred to be packaged in containers. This means a container is used as one type of resource to instantiate a PaaS service. As some other PaaS services prefer to use virtual machines to obtain an independent runtime environment, or prefer bare metal for better performance, a PaaS also uses virtual machines and bare metals to instantiate a PaaS service.
Roles	CSP, CSC
Figure (optional)	 **Figure I.12 – Utilization of service lifecycle management of PaaS**
Pre-conditions (optional)	The CSC selected a target PaaS service from the service catalogue of PaaS and subscribed to the PaaS service. The service catalogue triggered the lifecycle of the service after the subscription.
Post-conditions (optional)	
Derived requirements	– Service instantiation (refer to clause 8.4) – Service access control (refer to clause 8.4) – Service upgrading management (refer to clause 8.4) – Service instance deleting (refer to clause 8.4) – Service with multiple resource types (refer to clause 8.4)

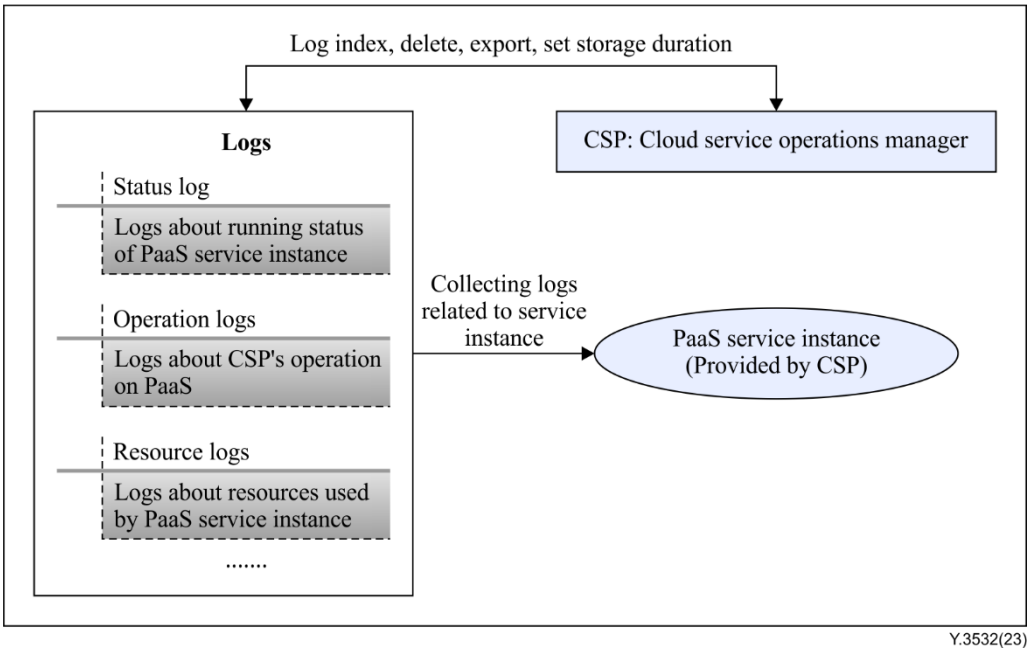
I.13 Use case: service metrics management of PaaS

Table I.13 – Use case: service metrics management of PaaS

Title	Service metrics management of PaaS
Description	This use case describes the PaaS CSP support to automatically monitor the PaaS service and corresponding resources which are subscribed to by the CSC. When a CSP provides a PaaS service to the CSC, the CSP is aware of the running status of the PaaS service itself and of the status of resources used by the PaaS service. These statuses are reflected by the metrics of PaaS service and the used resources. When the PaaS service is not working properly, the CSP checks the metrics to help with error detection. Monitoring metrics are stored, deleted or exported for future analysis. The storage duration would better support being defined by the CSP. The collected metrics beyond the duration of a health issue would be automatically deleted. A PaaS would better support customized alert rule settings based on metrics, so that when the running status of a PaaS service is not healthy, the CSP is alarmed and can take action.
Roles	CSP
Figure (optional)	 **Figure I.13 – Service metrics management of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Service metrics monitoring (refer to clause 8.4) – Service metrics management (refer to clause 8.4) – Service metrics storage duration management (refer to clause 8.4) – Alert on service metrics (refer to clause 8.4) – Service metrics report (refer to clause 8.4) – Metrics dashboard (refer to clause 8.4)

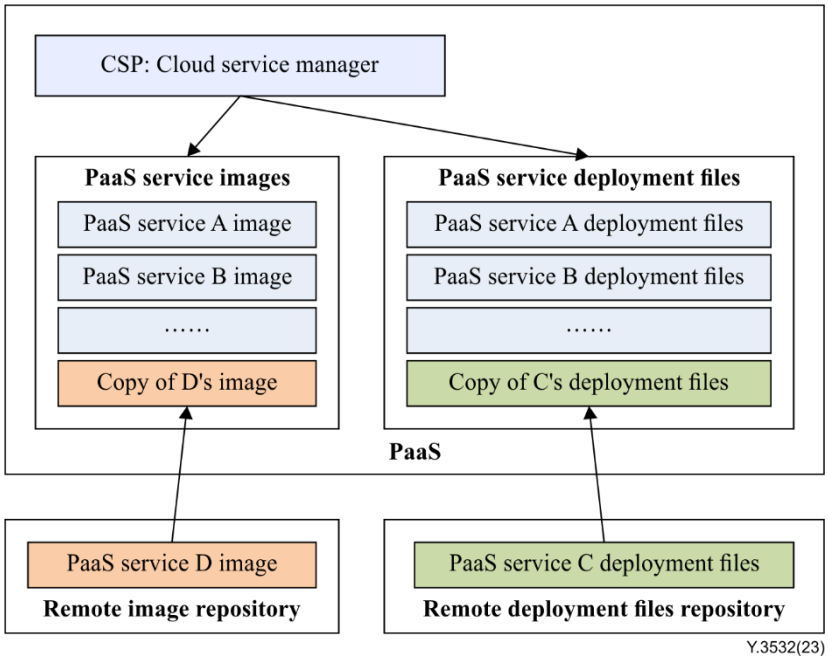
I.14 Use case: service log management of PaaS

Table I.14 – Use case: service log management of PaaS

Title	Service log management of PaaS
Description	This use case describes how the CSP supports automatically managing the log of PaaS and corresponding resources. It is common that a PaaS service is not working correctly, which requires troubleshooting by the CSP. Logs are an effective troubleshooting reference. So, for better reliability and faster troubleshooting, the CSP collects all the logs generated by PaaS services and corresponding resources, stores them for further analysis, deletes them if not useful, displays them if indexed and exports them if required. As a different PaaS service may need different log storage capacity, the CSP could provide a log storage duration setting. The logs beyond the duration would be automatically deleted.
Roles	CSP
Figure (optional)	 **Figure I.14 – Service log management of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Service log management (refer to clause 8.4) – Log storage duration management (refer to clause 8.4)

I.15 Use case: service image and deployment files management of PaaS

Table I.15 – Use case: service image and deployment files management of PaaS

Title	Service image and deployment file management of PaaS
Description	The PaaS service usually has two important files: service images and service deployment files. These two files are usually provided by the CSP and used to instantiate PaaS services after being subscribed to by the CSC. So, the CSP supports storing, updating, deleting images and deployment files of PaaS. As in cloud native era, there are a lot of open-source images and deployment files stored on the shared repository. The CSP may not acquire all images and deployment files locally. So, the CSP would also be able to access those public images and deployment file repositories to pull related files if not acquired locally.
Roles	CSP
Figure (optional)	 **Figure I.15 – Service image and deployment file management of PaaS**
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements	– Service image management (refer to clause 8.4) – Service deployment files management (refer to clause 8.4) – Pull remote image and deployment files (refer to clause 8.4)

Bibliography

[b-ITU-T Y.3501] Recommendation ITU-T Y.3501 (2016), *Cloud computing – Framework and high-level requirements*.

[b-ITU-T Y.4500.1] Recommendation ITU-T Y.4500.1 (2018), *oneM2M – Functional architecture.*

[b-ISO/IEC TR 13066] ISO/IEC TR 13066-6:2014, *Information technology – Interoperability with Assistive Technology (AT) – Part 6: Java accessibility application programming interface (API).*

[b-ISO/IEC/IEEE 24765] ISO/IEC/IEEE 24765:2017, *Systems and software engineering – Vocabulary.*

[b-ISO/IEC TS 23167] ISO/IEC TS 23267 (2020), *Information technology – Cloud computing – Common technologies and techniques.*

[b-NFV-IFA 029] ETSI GR NFV-IFA 029 (2019), *Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS".*

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Published in Switzerland
Geneva, 2023