



2019

**Guidance for
Software Conformity Certification**

APPLICATION OF “GUIDANCE FOR SOFTWARE CONFORMITY CERTIFICATION”

1. Unless otherwise noted, the requirements in the Guidance apply to software for which the application for software conformity certification is dated on or after July 1, 2019.

CONTENTS

CHAPTER 1	GENERAL	1
Section 1	General	1
CHAPTER 2	CONFORMITY CERTIFICATION	3
Section 1	Certification Application	3
Section 2	Review and Test	3
Section 3	Certification	4
CHAPTER 3	SOFTWARE DEVELOPMENT LIFE CYCLE	7
Section 1	General	7
Section 2	Planning	7
Section 3	Requirements	8
Section 4	Design and Implementation	9
Section 5	Validation and Verification	11
Section 6	Operation and Maintenance	13

CHAPTER 1 GENERAL

Section 1 General

101. Application

1. This guidance is applied to control, observation and safety system softwares installed on ships and/or offshore plants, operating support softwares for ships and/or offshore plants, product design and technical/engineering softwares for marine equipments.
2. Items not specified in this Guidance are to be in accordance with each relevant requirement in **Rules for the Classification of Steel Ships** (hereinafter referred to as "the Rules for Steel Ships") except for the requirements inapplicable to individual software products.
3. Items not included in this Guidance may comply with ISO, IEC or equivalent recognized standards by the appropriate consideration of the Society.
4. Where the specific requirements in international regulation such as IMO are or as Information technology & operating technology develops, when it deems necessary, additional requirements to this Guidance may be required.

102. Definitions

The definitions of terms are to follow the Rules for Steel ships, unless otherwise specified in this Guidance.

1. **"Software Product"** means a series of computer programs, procedures, related documents and data.
2. **"Requirements"** means an expression of the perceived needs of thing to be achieved or implemented.
3. **"Life cycle"** means a lifetime of the system from defining system requirements to the operation and maintenance of the system, which contains processes, activities and tasks related to the development, operation and maintenance of the software product.
4. **"System"** means a combination of elements that are constructed and interact to achieve one or more specific goals.
5. **"Verification"** means ensuring that the requirements specified by the provisions of objective evidence are fulfilled.
6. **"Configuration Item"** means processes, activities, and tasks related to the development, operation and maintenance of software products.
7. **"Configuration Management"** means management that presents technical and administrative direction to the design, development and support of configuration items including software items.
8. **"Baseline"** means the version of a formally approved configuration item that is officially designated and fixed at a particular time during the life cycle of the configuration item, regardless of media.
9. **"Configuration Control Board"** means a group that is responsible for accepting or rejecting changes in configuration items.
10. **"Interface"** means the boundary between hardware and software that make up a system or the two systems to be able to interact, or hardware, software, conditions, and conventions for inter-connecting at this boundary.
11. **"Bus"** means transporting pathway for data between internal and external sources.
12. **"Storage Devices"** means devices that store programs and data used by computers.
13. **"Integrity"** means ensuring that information in the software product is complete, accurate, legitimate and free from unauthorized changes.
14. **"Logging Process"** means a process of recording various kinds of information in operation to record and maintain the system operating state when operating the system, in order to perform an

investigation and analysis of system operation.

15. **"Architecture"** means a design of software in everything including a set of structures required to infer a system, which consists of elements and relationships at least.
16. **"Database"** means maintaining a set of data that is related to each other and has no redundancy in order to avoid duplication of data items and to unify information and perform processing efficiently.
17. **"Mechanism"** means the principle of operation of any software.
18. **"Application"** means software developed to perform a specific task.
19. **"Heap-based Memory"** means a memory space reserved for the operating system to allocate the memory space required during execution of the program.
20. **"Stack-based Memory"** means a reserved memory used for arithmetic calculations, local variables or to keep track of internal operation, through the way like the LIFO (Last-in First-Out).
21. **"Entry Point"** means an address in which a command to be executed first by a called program is stored because a program function is required when a program is being executed.
22. **"Garbage Collection"** means a form of memory management to reclaim memory no longer in use without manual specifying objects to deallocate and return to the memory system.
23. **"Premature Deallocation"** means that memory shall not be returned to the heap until there is a confirmation that it is not used elsewhere in the program.
24. **"Fragmentation Starvation"** means that when a program is loaded in a non-continuous place to maximize the utilization of the main memory device, a process is a case in which it does not get the privilege of the necessary resources forever and can not do its own work.
25. **"Call Graphs"** means a control flow graph, which represents calling relationships between sub-routines in a computer program.
26. **"Modularity"** means a characteristic of a software to be divided into small subsystems which interact with each other. Modules are divided based on functionality, and programmers are not involved with the functionalities of other modules. Thus, new functionalities may be easily programmed in separate modules.
27. **"Run-time Errors"** means an error in a library or program of a basic code used by a specific compiler or virtual machine to manage a program written in a computer language.
28. **"Commercial Off-The-Shelf"** means a product is usually a computer hardware or software product tailored for specific uses and made available to either a company or to the general public.
29. **"Simulator"** means a software that enables the execution of an application written for a different computer environment to imitate a real-world process or system.
30. **"Scenario"** means a description of the steps of general thought, including the interaction of the system environment with the user and the interaction between the components.
31. **"Regression test"** means a test used to determine whether changing part of an issue has created a new issue for a different part of the application.

103. Equivalence

The Society may consider the acceptance of alternatives to this Guidance, provided that they are deemed to be equivalent or above to those complying with the requirements of the Guidance.

104. Exclusion from the Guidance

The Society cannot assume responsibility for use of unauthorized commercial products and other technical characteristics not specified in the Guidance. ↓

CHAPTER 2 CONFORMITY CERTIFICATION

Section 1 Certification Application

101. General

1. Applicants for the software conformity certificate shall be submitted to the Society documents listed in **102.** and **103.** (three copies for review and one copy for reference) together with one copy of the application.
2. The documents shall be complied with requirements specified in **ISO 9001.**
3. The Society reviews the submitted documents in accordance with **Ch.3,** thereafter, the documents is sent to the applicant.

102. Documents review

1. Submission Documents
 - (1) Business performance plan
 - (2) Configuration management plan
 - (3) Requirements specification
 - (4) Basic and detailed design
 - (5) Source code or source code verification report
 - (6) Evaluation plan
 - (7) Evaluation result (including error correction report)
 - (8) Operation and maintenance plan

103. Documents for reference

1. Information on organization for software development
 - (1) Organization and management system for software development
 - (2) List of product, sales record
 - (3) Key developing person's career
2. User and/or operator manual, installation manual
3. Past test certification and/or report for acquiring the software conformity.
4. Other documents considered necessary by the Society

Section 2 Review and Test

201. Documents review

Data specified from **102.** and **103.** shall be submitted and approved in accordance with **Table 2.1** at each software life cycle respectively.

202. Certification test

1. The certification test shall be conducted after approval of the evaluation plan.
2. Certification tests shall be conducted in principle in the presence of the Surveyor. However, it may be omitted if it is carried out by the recognized.
3. The certification test shall consist of items to confirm the functions whether the requirements is operated properly.
4. The pass-fail testing shall be carried out for each items. The test shall not be carried out again without software modification in terms of fail items.

5. The items interconnected with other equipment shall be tested by actual or simulated environment. The simulated may be applied where it considers appropriate by the Society.
6. The applicant shall submit the evaluation result after completion of the test to the Society.

Table 2.1 Documents to submit per software development life cycle

Life Cycle	Submission Documents	Remarks
Planning	Business performance plan, configuration management plan	
Requirements Analysis	Requirement statement	
Design and Implementation	Basic and detailed design, Source code (or source code verification report)	
Software Verification	Evaluation plan, Evaluation result	Evaluation result shall be submitted after completion of the test.
Operation and Maintenance	Operation and maintenance plan	

Section 3 Certification

301. Certification notification

1. The Society shall issue a conformity certificate and send it to the applicant, when the evaluation results after the completion of certification test is satisfied.
2. The certificate shall state the conformity, scope and any additional restrictions and constraints of software products.
3. The software conformity certificate shall contain the following minimum information:
 - (1) Name and address of office where the software developer is registered
 - (2) Name and address of office where the supplier is registered (if different)
 - (3) Description, version number, and other related configuration information
 - (4) The environment in which the software is guaranteed to work(including other job restrictions or conditions)
 - (6) Certificate number and issue date
 - (7) Certificate validity period

302. Change of certification contents

1. The applicant shall notify any changes of the already certified software to the Society.
2. The applicant shall maintain and record contents of **302. 3.** in accordance with the software quality procedures related with the software version for all changes. The records shall be submitted upon a requisition by the Society. At this time, the Society reviews the change and may request a confirmation test as necessary.
3. The records shall be configured to track all changes as follows,
 - (1) Number of changes in project
 - (2) The version of the most recent software item, the release method identifier, the number of release and comparison between the release versions
 - (3) State of baseline
 - (4) Configuration control details(change control status)
 - (5) Configuration control committee activities
4. The following cases are not considered as a change and shall be carried out re-certification.
 - (1) Updated software at the already assessed(if expended)
 - (2) Updated installation and operation limit(if expended)
 - (3) Additional installation environment

303. Certification validity and reissue

1. The certificate validity period is three years from issue date. However, if the certificate is reissued in accordance with **302.**, the validity period shall be followed the existing issued date.
2. If there is no change in the component or structure, the Society may reissued the conformity certification without the tests.
3. Upon reissue of the conformity certification, additional information such as fault reporting and customer complaints during use of the software may be reviewed. additional testing may be required, if necessary. ↓

CHAPTER 3 SOFTWARE DEVELOPMENT LIFE CYCLE

Section 1 General

101. General

1. This guidance certifies conformity of software development life cycle.
 - (1) Planning
 - (2) Requirements
 - (3) Design and implementation
 - (4) Validation and verification
 - (5) Operation and maintenance
2. Software shall be developed and used in principle in English
3. The software shall be developed to carry out validation and verification activities per each a stage of life cycle.

Section 2 Planning

201. General

1. The business performance plan shall identify and describe the inputs, outputs, actors, and responsibilities of each activity per each a stage of life cycle.
2. The applicant or software provider shall designate a configuration manager. At this time, software development manager and The configuration manager may be the same.
3. Configuration management plan shall be submitted including the contents of **202.**

202. Configuration manager

1. The configuration manager implements the operating environment and carries out the configuration management activities according to the configuration management plan defined by project manager.
2. A role of configuration manager is as follows,
 - (1) Creating the configuration management plan
 - (2) Creating the configuration management environment and repository.
 - (3) Development and documentation of configuration management procedures
 - (4) Selection of baseline criteria
 - (5) Selection of the configuration identifier standard for files, documents, and versions
 - (6) Periodically report the configuration state

203. Configuration control

1. The configuration control committee consists of member as follows,
 - (1) The project manager
 - (2) Quality manager
 - (3) Technical manager and customer manager
2. A role of the configuration control committee is as follows,
 - (1) Determination of configuration items
 - (2) Determination of baseline
 - (3) Responsibility and warranty for reviewed changes
 - (4) Review any request for changes of baseline
3. Identified configuration items by the configuration management manager shall be managed by the following procedures.
 - (1) Request for changes

The customer or developer shall complete the change request and submit it to the configuration manager when the change occurs.

(2) Review for changes

The following shall determine whether a change request made by a customer or developer may be changed by the configuration control committee.

(a) Validity of reason for change(s)

(b) Scope for change(s)

(c) Impact for change(s)(duration, budget, manpower, etc.)

(3) Implementation for change

Change of configuration items is carried out after take the items in the repository

(4) Confirmation for change(s)

The configuration control committee shall review newly stored changes in the repository and the configuration management manager shall re-establish them as the baseline.

Section 3 Requirements

301. General

1. The requirements specification shall include activities related to acquisition, analysis, specification and validation and verification of functional and non-functional requirements.
2. Software requirements shall be divided into functional and non-functional requirements.
3. The software requirements shall be clearly specified per each a stage of software component.

302. Software requirements

1. The functional requirements for software are as follows,
 - (1) Operation
 - (2) Parameterization
 - (3) Definitions of inputs and outputs for software product (eg.. range and type of data, limits and default values)
 - (4) Interface
 - (a) Software systems and hardware systems (eg. data bus, drivers, target platforms, storage devices)
 - (b) Interfaces within software product
2. The non-functional requirements for software are as follows,
 - (1) Time and environment using software without error.
 - (2) Expected response time and storage periods
 - (3) Data classification levels, sensitive data management restrictions, authentication methods, permission levels, integrity checks, and logging processes
 - (4) Safety
 - (a) Alternative locations
 - (b) Redundancy strategy and error detection function
 - (5) Manual operation support and manpower interaction management
 - (6) Alerts, warnings and operator messages generated by the software
 - (7) Partition, use of discrete processors and other separation strategy requirements
 - (8) Installation and acceptance requirements for software versions delivered to development and maintenance sites
 - (9) The structure of the product or user documentation
 - (10) National or international standards
 - (11) Regulatory (if applicable)

303. Validation and verification of requirements

1. Software requirements shall be verified and recorded as follows,
 - (1) Completeness

The requirements shall be fully demonstrated without missing the necessary information.
 - (2) Unambiguity

- The requirements with clear expression shall be interpreted in the same sense.
- (3) Consistency
The requirements shall not be conflictive and inconsistent on the specification.
 - (4) Traceability
The requirements shall be traceable per each stage from the design to source code.
 - (5) Easy modifiability
The requirements shall be described independently to avoid inconsistencies or disagreements.
 - (6) Verifiability
The requirements shall be systematically verified without ambiguity in the sentence.

Section 4 Design and Implementation

401. General

1. The software design shall be carried out taking into account as follows,
 - (1) The basic and detailed design documents shall include all of the requirements specifications.
 - (2) The design shall be traceable for easy maintenance.
 - (3) The design shall be adaptable to change easily.
 - (4) The design shall minimize the impact of system changes.
 - (5) The design shall be easily read and understood.
2. The design and implementation shall define the software architecture through the software requirements of **302**.
3. The software architecture shall identify interface between software components.
4. The basic and detailed design shall be carried out in design and implementation. However, it may be omitted when the basic design is not needed separately.

402. Design requirements

1. Software developers shall define and record the design requirements in addition to the software as requirements of **302**. follows.
 - (1) Software development language and compatibility software components
 - (2) Timing and memory characteristics of operating system and network infrastructure
 - (3) Range and type of data, limits and default values
 - (4) Interface description and external interface description between software components
 - (5) Alarms, warnings, and operator messages generated by each software component
 - (6) Naming, formatting rules and conversion functions
 - (7) Function, performance, security, safety and fault isolation requirements assigned to the component
2. The requirements of 1. shall be verified and recorded as follows,
 - (1) The software architecture shall be consistent.
 - (2) The requirements shall not conflict with each other.
 - (3) The requirements shall be expressed in clear terms.(The requirements may use formal methods.)
 - (4) The design requirements may be a basis for validation and verification activities.
 - (5) The design requirements may be uniquely identified.
 - (6) The design requirements shall identify source of the software requirements or other.

403. Architecture

The architecture shall include the features and functions as follows,

- (1) The overall structure of the software to be developed shall be addressed.
- (2) The architecture shall address the various components(subsystems, components) that make up the software.
- (3) The architecture shall define the interaction between components over the interface.
- (4) The architecture shall address more important than details.
- (5) The architecture shall have standard applicable to the design and development of the system.
- (6) The architecture shall be utilized as a communication tool.

- (7) constraints for implementation shall be defined.
- (8) Quality attributes shall be determined.
- (9) The architecture shall be designed to be reusable.

404. Basic design

1. The basic design defines the architecture, application, interface, and database.
2. The developer shall describe all interfaces between software components and external components(eg. other system software and hardware systems). At this time, the external interface includes a bus, a storage device(eg. RAM, EEPROM, FLASH, etc.) and an input/output device usable in a software system.
3. The architecture shall specify the operation of the software system including the following items for the timing requirements.
 - (1) Priority among data, maximum execution time and activation conditions
 - (2) Synchronization and scheduling of different input points
 - (3) Protection mechanisms for shared variables(and legitimacy of interlocking)
 - (4) Identification of shared variables
4. The entry point for a software component shall request a work, interruption or subject of the software system.
5. The architecture shall specify the operating of the software system for the memory management requirements as follows,
 - (1) Heap-based or stack-based memory allocation, garbage collection, etc.
 - (2) Lack of heap memory, early release of memory or fragmentation starvation
6. The protection mechanism shall be defined by the maximum depth of the call graphs.

405. Validation and verification of basic design

Basic design shall be verified and recorded as follows,

- (1) The software architecture shall implement software requirements.
- (2) The software architecture shall be consistent in modularity.
- (3) The software architecture shall describe all interfaces between software systems.

406. Detailed design

1. Each software components of the software architecture shall be divided into the appropriate unit software.
2. Detailed design shall be developed for units of the software components. And it shall be defined as the actual inputs and outputs of each unit software including a type and functional area.
3. The detailed designs shall be developed for interfaces between unit software and external components (hardware and software).

407. Verification of detailed design

The software detailed design shall be verified and recorded as follows,

- (1) Software components
- (2) Software architecture
- (3) Design criteria
- (4) Software consistency

408. Implementation

1. Each unit of software shall be implemented for the basic design and detailed design.
2. The source code shall be separately defined and applied as follows,
 - (1) Naming standard.
 - (2) Style convention (file, header, function, annotation of code, etc.)
 - (3) Quantitative goals (complexity, size of function, etc.)

- (4) Environment constraints (compiler options, etc.)
- (5) Programming language restrictions (prohibiting or controlling the use of dangerous or complex configurations)
3. Dead code (unnecessary code or unused code) and runtime errors in the source code shall be detected. Recursive phrases are allowed only when they are controlled by the maximum number of iterations.
4. The accuracy of the figures shall be expected and verified.

409. Implementation verification

1. The source code shall be verified and recorded as follows,
 - (1) The source code of the software shall conform to the defined programming procedures and/or coding standard.
 - (2) The inconsistent parts of the implementation results shall be analyzed and corrected.
 - (3) The source code shall be made without mistake according to the software requirements and design requirements.
 - (4) The logic of the source code shall be clearly and not unnecessarily complex.
 - (5) The logic of the source code shall be easy to read and well annotated. Annotation processing of commercial products shall describe information for integration, register and composition.
 - (6) The program shall not have unexpected bottlenecks for performance.
2. The verification of the source code of 1. may be carried out by the recognized by the Society. In this case, the source code verification report shall be submitted to the Society and reviewed.
3. Code analysis tools may be used to identify dead code and run-time errors.

Section 5 Validation and Verification

501. General

The implemented software shall verify to satisfy with the requirements. The hardware shall be installed when the software is embedded in the hardware.

502. Software Evaluation Plan

1. The software evaluation plan shall include plans, methods, and procedures for evaluating the suitability of the software as follows,
 - (1) All features in the product description and user description
 - (2) Test purpose, input data, expected results and pass-fail criteria
 - (3) Test procedure that includes preparation for testing and recording results
2. The test shall be carried out in at least two cases for each function and feature of the software.
3. The test shall be carried out for installation and operating restrictions as indicated in the product's documentation and user documentation.
4. The test plan shall specify the testing hardware and software configurations, including all configurations mentioned in the software product description. All applied tools for software testing shall be specified.
5. Validation strategies shall define the test bench required to reach the activity goals. Also, The simulator may also be that the software system is tested in the real environment.
6. The interaction of hardware and/or software components shall be considered when testing the software in the evaluation plan.
7. The procedure shall be established for retesting the relevant function or feature if the test result does not meet the criteria.

503. Software Verification Criteria

The acceptance criteria for software shall be verified as follows,

- (1) Expected requirements under normal conditions
- (2) Unexpected requirements under normal conditions
- (3) Expected requirements under abnormal conditions
- (4) Unexpected requirements under abnormal conditions
- (5) Conditions that may lead to errors that were previously identified

504. Software Verification

1. The conformity of the software components shall be verified in accordance with evaluation plan. This activity shall be carried out in the final goal environment.
2. All verification procedures and results (pass-fail as well as discrepancies) shall be documented and preserved.
3. The documentation of the evaluation results shall be considered as follows,
 - (1) Tracking software requirements and designs
 - (2) Software requirements and consistency of design
 - (3) Consistency between design requirements
 - (4) Test range of software
 - (5) Possibility of operation and maintenance
4. The software used in the test shall be the same as the software under conformity evaluation.

505. Evaluation Result Sheet

The evaluation result sheet shall include information as follows,

- (1) All simulation designs, simulation scenarios, simulation procedures and simulation results
- (2) Proof of all evaluation to be carried out according to plan
- (3) A reference to the evaluation date, the name and function of the testing device, the list of identified exceptions and the corresponding exception reports
- (4) Report on each evaluation case

506. Retesting

1. The evaluation results shall be documented if they do not meet the reviewed criteria as follows,
 - (1) Specific items identified by the evaluation and location and characteristics of test cases
 - (2) Modification of specific items and retesting verification plan
 - (3) Items shall be further described for correction and demonstration of reflection result as follows.
 - (a) Identifier of the modification
 - (b) Date of modification
 - (c) Name of the modifier
 - (d) Identifier of the change corresponding to the modification
 - (e) The impact of the modification
 - (f) Comments of the modification
 - (4) The relevant data shall be submitted to the Society for review if it is not necessary to modify the specific items.
2. The software shall modify the identified specific items to ensure that the verification criteria defined in **503.** are met.
3. The software functions shall be further described in the evaluation results for which they have been re-tested as follows,
 - (1) The identifier of the verification
 - (2) The date of verification
 - (3) The name of the verifier
 - (4) The test cases used for the verification
 - (4) The Types of verification
 - (5) Results of the verification

Section 6 Operation and Maintenance

601. General

1. Whole activities of software product for operation and maintenance shall be performed in accordance with the requirements.
2. Software product shall be replicated, delivered and maintained under the quality management system.
3. Software product shall be maintained systematically by the way through the configuration management system.

602. Operation and Maintenance

1. Quality control

A quality control plan for a software product is that it identifies and describes the responsibilities and authorities involved in implement and validation the configuration process through all stages of the software development life cycle. The interface shall be defined between different activities related to the configuration management. And the responsible authority shall also be identified for verifying implementation activities.

2. Replication and delivery

Replication records for the software shall be including, but not limited to, as follows,

- (1) The master and copies, including format, variant and version.
- (2) The used media type and associated labelling
- (3) Instruction and user manual, licences, release records of the software product including an identifier and composition
- (4) The verification of the correctness and completeness of delivered copies of the software product
- (5) Protection measures for the software product against damage during delivery

603. Maintenance

1. The maintenance for the software product is as follows,

(1) Modification maintenance

Errors of the software product shall be corrected though those are found after delivery.

(2) Adaptation maintenance

Developed software product shall be modified or supplemented corresponding to changes in environment such as operating system, peripherals, network environment, etc.

(3) Functional reinforcement maintenance

Developed software product shall perform functional reinforcement maintenance when it requires the addition, modification, and performance improvement of new functions.

(4) Preventive maintenance

Errors shall be found by prediction and expectation rather than not to correct after finding problems.

2. The software product shall be planned and managed to keep maintenance records are maintained and at least include as follows,

- (1) The list of reported problems and their current status
- (2) The authority responsible for implementing corrective action
- (3) The priorities assigned to corrective actions
- (4) The results of corrective actions
- (5) The methods of notice to customers for planned future changes

604. Change Management

1. Each change requests shall be recorded and analyzed.
2. After the change plan has been established, the impact of the release version of the software shall be analyzed. The impact of the release version of the software shall be analyzed after establishing the change plan. A plan shall be established for the software development activities that need to be repeated during the impact analysis and for the documents to be updated.

3. When modifying the release version of the software, a regression test shall be performed to ensure that there are no new errors introduced by the change.
4. The scope of the validation and verification activities shall be identified according to the type of software changes.

605. Configuration Audit

1. Configuration audit is conducted by the configuration management manager and a plan for the audit shall be established on the configuration management plan.
2. The configuration audit shall be verified with at least the following contents.
 - (1) Verification of the reflection of approved change requests
 - (2) Verification of non-approved reflections
 - (3) Verification of renewal of items related to approved changes

606. Software Release

The software version shall be released as follows,

- (1) Ensure that all the verification activities have been done;
- (2) Each existing exceptions in the software product shall be documented.
- (3) Version of the software product shall be uniquely identified.
- (4) All outputs related to the software product, such as documents, model files, test results, source code, etc., shall be collected and identified.
- (5) Procedure to install the software product shall be documented.
- (6) Procedure shall be documented to test the software product at final environment after installation.
- (7) Procedure shall be documented to get feedback from the operational field.

607. Operation and Maintenance Plan

1. Document such as operational manuals shall be included in operation and maintenance plan.
2. The operation and maintenance manager of the software shall be designated. ↓

GUIDANCE FOR SOFTWARE CONFORMITY CERTIFICATION

Published by

KR

36, Myeongji ocean city 9-ro, Gangseo-gu,
BUSAN, KOREA

TEL : +82 70 8799 7114

FAX : +82 70 8799 8999

Website : <http://www.krs.co.kr>

Copyright© 2019, **KR**

Reproduction of this Guidance in whole or in parts is
prohibited without permission of the publisher.